H2 Database Engine Documentation 《H2 Database 教程》

waylau

Published with GitBook



目錄

1. 介紹
2. 入门
3. 安装
4. 教程
    i. H2 控制台
    ii. 使用数据库
    iii. 数据库升级、备份、恢复
    iv. 工具
    v. 用户定义变量
    vi. 日期和时间
    vii. 与第三方集成
5. 特性
6. 性能
7. 高级

H2 Database Engine Documentation 《H2 Database 教程》



It is a book about the H2 Database Engine. Through this book, you can quickly start with H2. This is a GitBook version of the book: http://waylau.gitbooks.io/h2-database-doc. The current version of H2 is 1.4.188 (2015-08-01), Beta. Let's READ!

《H2 Database 教程》是关于 H2 Database Engine 的一个中文教程。H2 是一款短小精干的 Java 内存数据库,性能强劲。至今为止，H2 的最新版本为 Version 1.4.188 (2015-08-01), Beta。插入配图， 图文并茂方便用户理解， 带你快速掌握 H2。本书利用业余时间编写,由于时间紧凑,精力和能力有限,书中未免有纰漏和错误,望读者能够热忱斧正。可以在https://github.com/waylau/h2-database-doc/issues提供意见。

另外有 GitBook 的版本方便阅读http://waylau.gitbooks.io/h2-database-doc.

https://github.com/waylau/h2-demos 是关于使用 H2 的例子。

从目录开始阅读吧

Contact:

- Blog:www.waylau.com
- Gmail: waylau521@gmail.com
- Weibo: waylau521
- Twitter: waylau521
- Github : waylau

入门

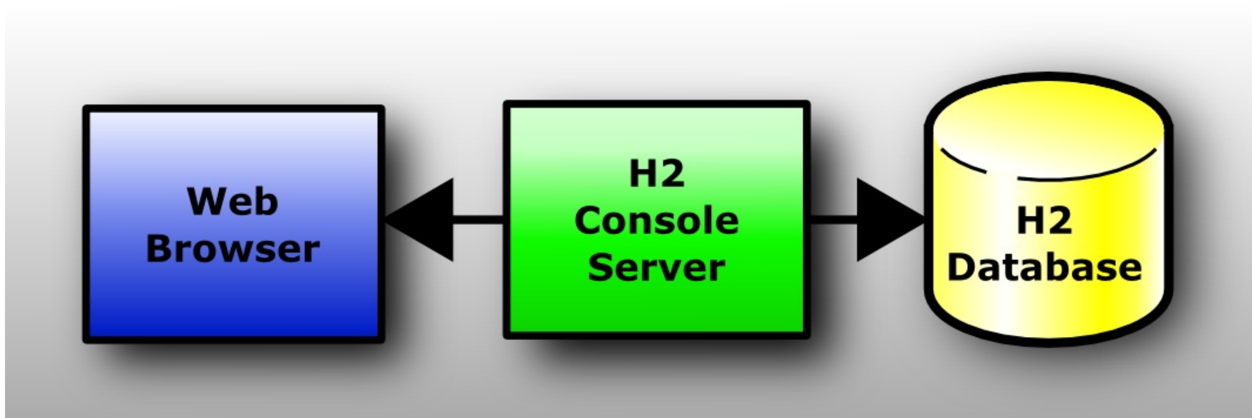
内嵌 H2 在应用中

H2 可以使用内嵌模式或者是服务器模式。使用内嵌模式，需要做如下步骤：

- 添加 `h2*.jar` 到 classpath (H2 没有任何依赖)
- 使用 JDBC 驱动类：`org.h2.Driver`
- 数据库的 URL 是 `jdbc:h2:~/test`,在你本地目录打开数据库 `test`
- 此时，新的数据库就自动创建了

使用 H2 控制台

控制台运行你在浏览器访问 SQL 数据库。



如果没有正常运行，参见教程)一章。

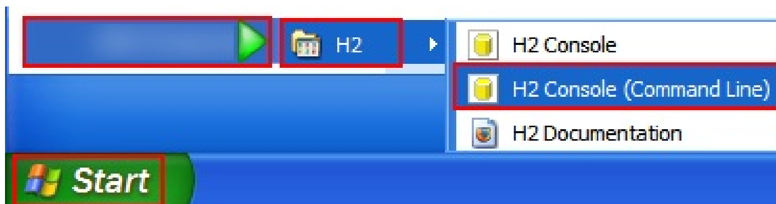
步骤

安装

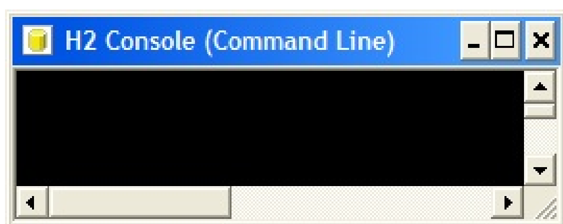
使用 Windows Installer 安装软件

启动控制台

点击 [Start], [All Programs], [H2], and [H2 Console (Command Line)]:



出现控制台窗口:



同时浏览器会弹出一个新页面http://localhost:8082。可能会有防火墙的安全警告。如果是在本地访问数据库，就不用处理。

登陆

选择 [Generic H2]，点击 [Connect]:

English Preferences Help

Login

Saved Settings: Generic H2

Setting Name: Generic H2 Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:test

User Name: sa

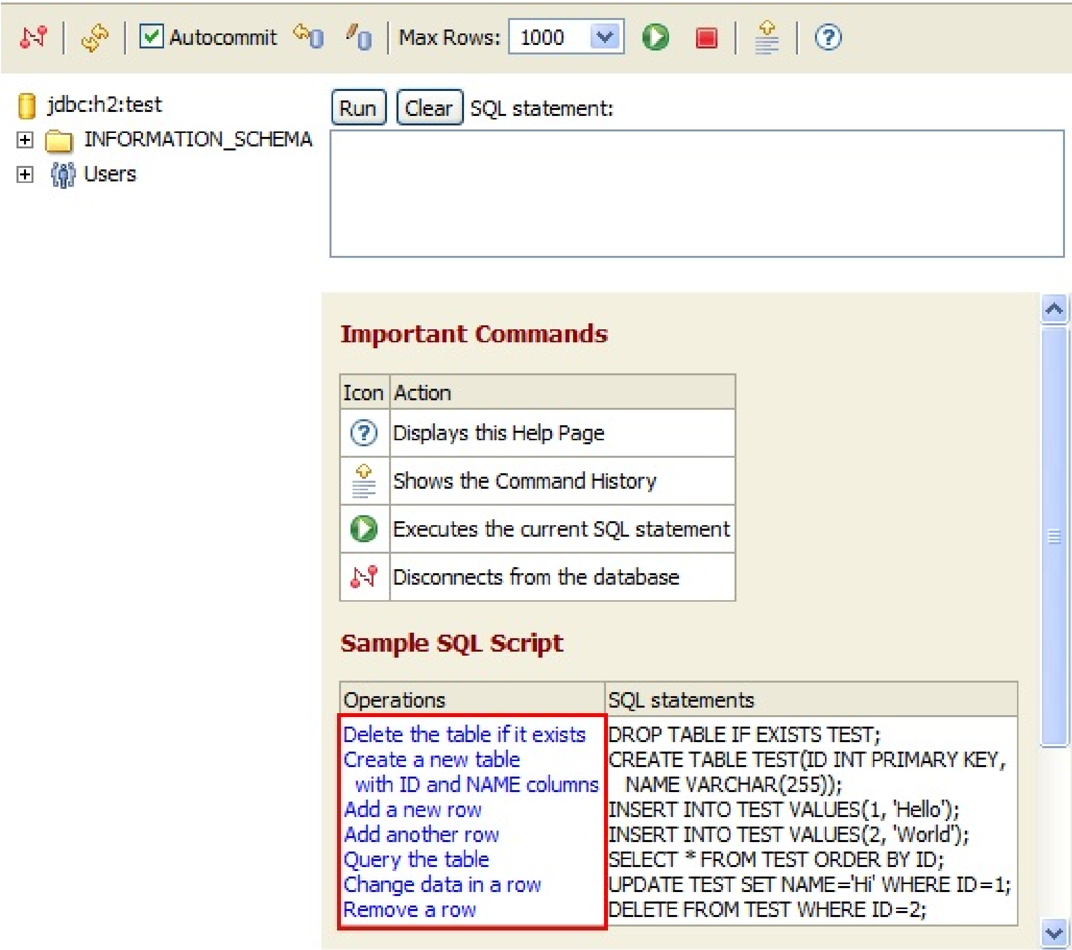
Password:

Connect Test Connection

现在可以登陆了

示例

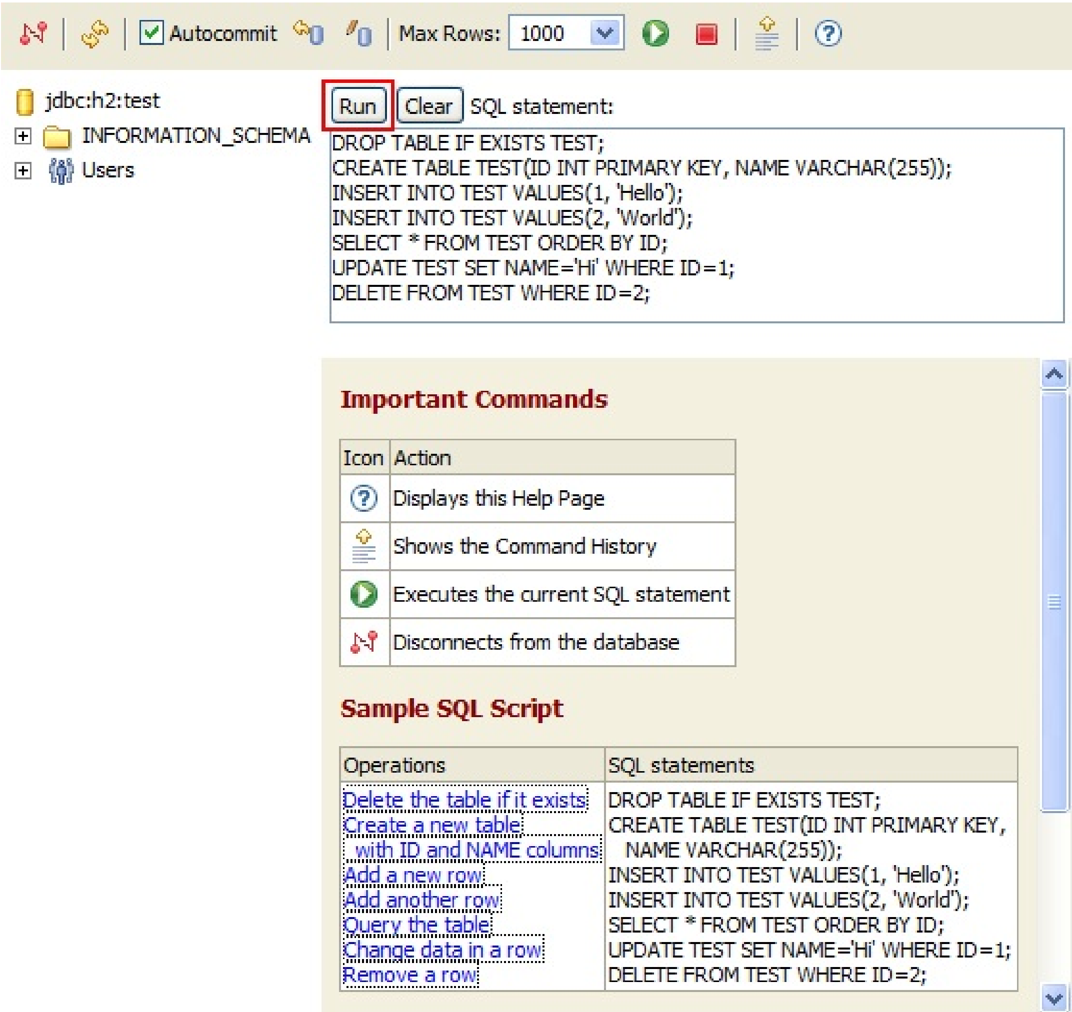
点击 [Sample SQL Script]:



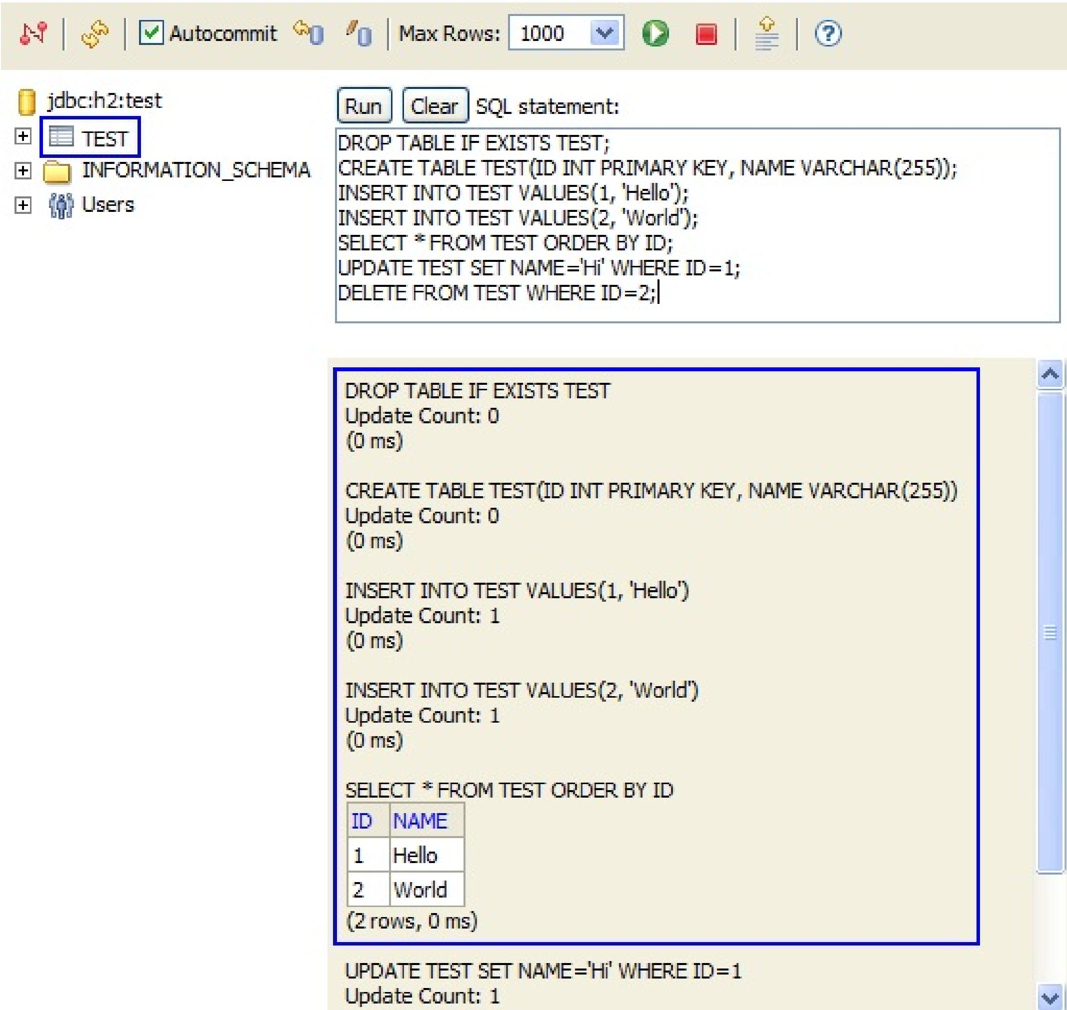
SQL 命令就出现在命令输入区

执行

点击 [Run]



在左侧，一个的条目 TEST 就添加到了数据库图标下。操作和结果是显示如下脚本



断开连接

点击 [Disconnect]:



来关闭连接

结束

关闭控制台窗口。详见教程一章。

安装

环境

运行本数据库，需要安装下面软件。其他软件可能也可以，但没有测试过。

数据库引擎

- Windows XP 或者 Vista, Mac OS X, 或者 Linux
- Sun Java 6 或更新
- 推荐用 Windows 文件系统： NTFS (FAT32 只支持文件最大为 4 GB)

H2 控制台

- Mozilla Firefox

支持的平台

本数据库是Java 编写的，故可以跨平台使用。已经在 Java 6 和 7 上做了测试。当前，数据库的开发和测试已经在使用了 Java 6 的 Windows 8 和 Mac OS X，但仍可以运行在其他系统和其他的 Java 。支持所有的主流操作系统(Windows XP, Windows Vista, Windows 7, Mac OS, Ubuntu,...)

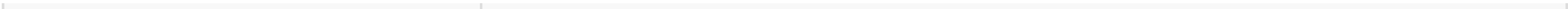
安装软件

运行 installer 或者解压安装包到任意目录

目录结构

安装后，目录结构如下：

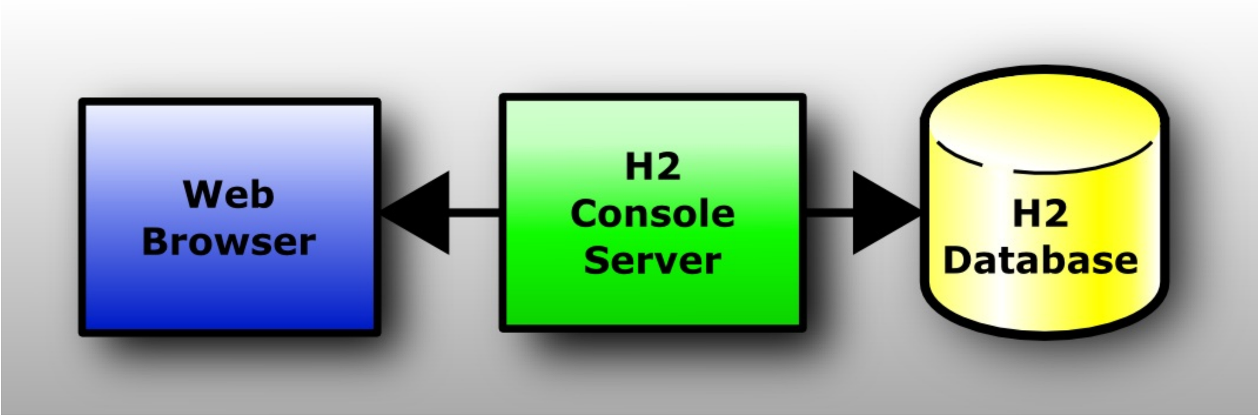
Directory	Contents
bin	JAR 和批处理文件
docs	文档
docs/html	HTML 页面
docs/javadoc	Javadoc 文件
ext	外部依赖 (构建时下载的)
service	运行数据库作为 Windows 的 Service
src	源文件
src/docsrc	文档源文件
src/installer	安装程序,shell和发布的构建脚本
src/main	数据库引擎源代码
src/test	测试源代码
src/tools	工作和数据库适配器源代码



教程

使用和启动H2管理系统

H2 管理系统让你能够通过一个浏览器对 H2 的 SQL 数据库进行管理操作。H2 管理系统不仅可以连接 H2 数据库，也可以连接其他支持 JDBC API 的数据库。



这是一个 C/S 应用，在服务器和客户端（浏览器）上都要运行 H2 的管理程序。根据平台不同，H2 管理系统支持多种启动应用的方式：

操作系统	启动
Windows	点击 [Start], [All Programs], [H2], 和 [H2 Console (Command Line)]。系统的系统托盘可以看到 H2 的图标。如果看不到可能是 Java 没有安装正确。浏览器访问http://localhost:8082。
Windows	打开文件浏览器，切换目录到h2/bin，双击运行h2.bat。一个控制台窗口将弹出，如果有问题，将有错误信息在这个窗口里显示。一个浏览器窗口将被打开，指向的URL是http://localhost:8082，是H2管理系统的登录页面。
Any	双击h2*.jar文件。前提是 .jar 文件能正确的被 Java 打开。
Any	开启控制台窗口，切换目录到h2/bin，执行命令： `java -cp h2*.jar org.h2.tools.Server`

防火墙

在你启动服务时，如果你安装了防护墙，你可能会收到一个防护墙的安全警告。如果不需要其他计算机访问你这台计算机上的H2数据库，你可以让防火墙阻塞H2对外服务的端口，但是本地计算机仍可以访问这些端口。当你需要其他计算机也能访问这台计算机的H2数据库时，你需要让防火墙开放H2对外服务的端口。

有报告显示使用 Kaspersky(卡巴斯基)7.0 的防火墙时，使用 IP 地址访问本地的 H2 时，速度非常的缓慢，替代的方案是使用'localhost'代替IP 地址来访问。

一个简单的防火墙已经集成到H2的服务器中，其他的计算机缺省状态下不能连接到服务器，如果需要其他计算机能连接到H2服务器，到'Preferences'（偏好），选择'Allow connections from other computers'（允许从其他计算连接）即可

JAVA测试

打开一个命令行窗口，输入下面的命令，检测JAVA的版本：

```
java -version
```

如果你得到错误的信息，你可能未安装 JDK，或是需要将 Java 的可执行文件路径加入到环境变量 PATH 中。

错误信息'Port may be in use'(端口被占用)

你可能在启动一个 H2 控制台实例时，出现错误信息"The Web server could not be started. Possible cause: another server is already running...".（WEB服务器不能启动，可能的原因：另外一个服务器已经在运行了）。使用不同的端口，可以在一台计算机上启动多个控制台程序，但是一般都不会做。

使用其他端口

如果端口已经被其他应用占用，你需要使用其他端口来启动 H2 控制台。改变 H2 的控制台端口需要修改配置文件 `.h2.server.properties`。这个文件存储在用户目录下(在 Windows 系统中，这个文件通常在 `Documents and Settings/<username>`)。这个相应的入口实体是 webPort.

使用浏览器连接到服务器

服务器启动成功后，你就可以使用 WEB 浏览器访问服务，浏览器需要支持JavaScript。在启动的服务器上启动浏览器，打开 URL http://localhost:8082。在启动服务器之外的计算机上，你需要提供启动服务器的IP地址，如http://192.168.0.2:8082.如果你在服务器上启用了SSL， URL需要使用https://开头.

多个并发会话

支持多个并发的浏览器会话。由于数据对象是存储在服务器上的，同时工作的会话数受限于服务器的内存。

登录

在登录页，你提交连接信息就可以登录到数据库。设置 JDBC 作为连接数据库的驱动，填入JDBC URL、用户名、密码，单击[Connect]。你能保存和恢复以前设置的信息，这些设置都存储在属性文件中。

错误信息

错误信息用红色标识，你能通过单击消息显示或隐藏异常的堆栈信息。

附加数据库驱动

通过增加 jar 的本地驱动文件到环境变量来附加数据库驱动 (MySQL, PostgreSQL, HSQLDB,...)。环境变量包括 H2DRIVERS 和 CLASSPATH，以 Windows 为例：如要增加数据库驱动 `C:\Programs\hsqldb\lib\hsqldb.jar`，设置环境变量 H2DRIVERS 为 `C:\Programs\hsqldb\lib\hsqldb.jar`

多个驱动可以被设置，每个驱动之间通过 `;` 分号分隔（Windows），其他系统通过 `:` 冒号分隔。在路径中空格被支持，但是这些设置不能被引用。

使用 **H2** 控制台

H2 控制台主页面分为三个主要的部分：顶部的工具栏，左边的是对象树，右边的是查询和结果输出栏。数据库对象（如表）都被列在左边的树形上。在查询栏上输入SQL语句点击 [Run]，结果就被输出到命令行的下面。

增加表名和字段名

可以通过在树上点击增加表名和字段名到脚本。如果单击表，当这个查询栏是空的时候， `SELECT * FROM...` 将被自动增加到查询栏。当输入一个表的查询时，对象树上将自动的展开这张表。例如，你输入 `SELECT * FROM TEST T WHERE T`，对象树上的表 TEST 将自动的展开

断开连接和停止应用

断开数据库，点击工具栏上的 [Disconnect] 即可，这个时候，数据库服务仍在继续运行，等待着一个新的会话进行连接。

停止服务需要右键点击系统托盘的 H2 图标，选择 [Exit]。如果没有系统托盘的 H2 图标，切换到 [Preferences] 单击 [Shutdown]，在Windows 上在服务器启动的窗口下按[Ctrl]+[C]，或者直接关闭 Windows 上的控制台窗口。

H2 控制台的特殊语法

H2 控制台支持几个内置命令。这些都在 H2 控制台做解析,所以他们使用任何数据库。内置命令需要在声明的开始(在任何备注之前),否则他们是无法正确解析。如果有疑问,之前添加 `;` 命令。

Command(s)	Description
@autocommit_true; @autocommit_false;	Enable or disable autocommit.
@cancel;	Cancel the currently running statement.
@columns null null TEST; @index_info null null TEST; @tables; @tables null null TEST;	Call the corresponding `DatabaseMetaData.get` method. Patterns are case sensitive (usually identifiers are uppercase). For information about the parameters, see the Javadoc documentation. Missing parameters at the end of the line are set to null. The complete list of metadata commands is: `@attributes, @best_row_identifier, @catalogs, @columns, @column_privileges, @cross_references, @exported_keys, @imported_keys, @index_info, @primary_keys, @procedures, @procedure_columns, @schemas, @super_tables, @super_types, @tables, @table_privileges, @table_types, @type_info, @udts, @version_columns`
@edit select * from test;	Use an updatable result set.
@generated insert into test() values();	Show the result of `Statement.getGeneratedKeys()` .
@history;	List the command history.
@info;	Display the result of various `Connection` and `DatabaseMetaData` methods.
@list select * from test;	Show the result set in list format (each column on its own line, with row numbers).
@loop 1000 select ?, ?/*rnd*/; @loop 1000 @statement select ?;	Run the statement this many times. Parameters (`?`) are set using a loop from 0 up to x - 1. Random values are used for each `?/*rnd*/` . A Statement object is used instead of a PreparedStatement if `@statement` is used. Result sets are read until `ResultSet.next()` returns `false` . Timing information is printed.
@maxrows 20;	Set the maximum number of rows to display.
@memory;	Show the used and free memory. This will call `System.gc()` .
@meta select 1;	List the `ResultSetMetaData` after running the query.
@parameter_meta select ?;	Show the result of the `PreparedStatement.getParameterMetaData()` calls. The statement is not executed.
@prof_start; call hash('SHA256', '', 1000000); @prof_stop;	Start/stop the built-in profiling tool. The top 3 stack traces of the statement(s) between start and stop are listed (if there are 3).
@prof_start; @sleep 10; @prof_stop;	Sleep for a number of seconds. Used to profile a long running query or operation that is running in another session (but in the same process).
@transaction_isolation; @transaction_isolation 2;	Display (without parameters) or change (with parameters 1, 2, 4, 8) the transaction isolation level.

设置 H2 控制台

H2 控制台的设置信息存储在配置文件 `.h2.server.properties`，该文件存放在你的用户目录下。在 Windows 上，用户目录通常是 `C:\Documents and Settings\[username]` 或者 `C:\Users\[username]`。H2 控制台第一次启动时将自动创建应用所需要包含的配置文件。 支持的配置如下：

- webAllowOthers: 允许其他电脑连接
- webPort: H2 控制台的端口
- webSSL: 使用加密的 TLS (HTTPS) 连接 * 除了这些设置,列出了最近使用连接的属性的格式 `<number>=<name>|<driver>|<url>|<user>` 使用转义字符 `\`。举例 `1=Generic H2 (Embedded)|org.h2.Driver|jdbc\:h2\:~/test|sa`

使用 JDBC 连接数据库

JAVA应用要连接到数据库，首先需要加载数据库驱动，然后获得一个数据库连接，下面是一个简单的例子：

```
import java.sql.*;
public class Test {
    public static void main(String[] a)
            throws Exception {
        Class.forName("org.h2.Driver");
        Connection conn = DriverManager.
            getConnection("jdbc:h2:~/test", "sa", "");
        // add application code here
        conn.close();
    }
}
```

代码中通过 `Class.forName(...)` 来加载驱动，通过 `DriverManager.getConnection()` 来打开一个连接，驱动名为"org.h2.Driver"。数据库 URL 总是使用 `jdbc:h2:` 来标识， `getConnection()` 的第二个参数是用户名（ `sa` 作为系统超级管理员的一个例子），第三个参数是密码，用户名是不区分大小写，但是密码是大小写区分的。

创建新一个新数据库

缺省情况下，如果 URL 指定的数据库并不存在，一个新的空的数据库将被自动的创建。创建数据库的用户自动成为这个数据库的超级管理员。

自动创建新库也可以通过特殊的 URL 进行屏蔽，参见打开一个存在的数据库。

使用服务器模式

H2 目前支持三种服务器模式：web 服务器模式（H2 控制台）、TCP 服务器模式（C/S连接）和 PG 服务器模式（PostgreSQL 客户端）。可以通过多种方式启动服务器模式，通常的方式是通过 `Server` 工具。 开启服务器无需开启数据库——数据库会在客户端连接时开启

通过命令行启动 Server 工具

缺省设置下，输入下面命令并执行能启动 Server 工具：

```
java -cp h2*.jar org.h2.tools.Server
```

通过下面的命令行，可以查看服务器启动命令行的参数及缺省值：

```
java -cp h2*.jar org.h2.tools.Server -?
```

参数允许服务器工具启动到其他端口或者只是部分启动。

连接到 TCP 服务器

为了能远程通过 TCP 服务器访问数据库，使用下面的连接驱动和数据库URL：

• JDBC驱动类： `org.h2.Driver` • 数据库URL: `jdbc:h2:tcp://localhost/~/test`

关于数据库 URL，可以查看《特性》这章。注意不能通过浏览器访问这个 URL，只能通过 H2 客户端（通过 JDBC）。

在应用内部启动TCP服务

在JAVA应用内部，也可以通过代码来实现TCP服务的启动和停止，例子代码如下：

```
import org.h2.tools.Server;
...
// 启动 TCP Server
Server server = Server.createTcpServer(args).start();
...
// 关闭 TCP Server
server.stop();
```

从其他程序关闭 TCP 服务器

可以从另外的程序关闭 TCP 服务器，使用下面命令行：

```
java org.h2.tools.Server -tcpShutdown tcp://localhost:9092
```

在用户应用中关闭服务器，使用:

```
org.h2.tools.Server.shutdownTcpServer("tcp://localhost:9094");
```

这个功能将仅仅关闭 TCP 服务器。如果相同的程序有其他服务器，他们不会被关闭，而是继续执行。为了避免覆盖在数据库下次打开时，在调用这个方法时，所有的到数据库的连接将会关闭。要实现远程关闭服务器，需启用远程连接。关闭一个 TCP 服务器可以通过选项 `-tcpPassword` 来保护 (启动和关闭 TCP 服务器也要用这个密码)

使用 Hibernate

H2 数据库支持 Hibernate 3.1及以上的版本。 你能够使用 HSQLDB 方言，或是 H2 自己的方言。注意的是，在 Hibernate 中包含的 H2 方言有BUG，针对这些 BUG 的补丁已经被发布和修复,见https://hibernate.atlassian.net/browse/HHH-3401。你能够将它改名为H2Dialect.java，直接把它包含在你的应用中即可使用，或者使用 已经修复了这个问题的 Hibernate 的版本。

当使用 Hibernate,尝试使用 `H2Dialect` 如果可能的话。当使用 `H2Dialect` ,兼容性模式比如 `MODE=MySQL` 是不支持的。当使用兼容模式时,使用 Hibernate 相应的数据库的方言，而不是 `H2Dialect` ;但请注意 H2 不支持所有数据库的所有功能。

使用 TopLink 和 Glassfish

在 Glassfish （或 Sun AS）中使用 H2，设置 Datasource Classname 为 `org.h2.jdbcx.JdbcDataSource` 。可以通过图形界面进行设置[Application Server] - [Resources] - [JDBC] - [Connection Pools], 或者编辑文件 `sun-resources.xml` ：修改元素 `jdbc-`

`connection-pool`，设置属性 `datasource-classname` 为 `org.h2.jdbcx.JdbcDataSource`。

H2 数据库是兼容 HSQLDB 和 PostgreSQL。如果要使用 H2 的特殊属性，需要使用 `H2Platform`，源代码在 `src/tools/oracle/toplink/essentials/platform/database/DatabasePlatform.java.txt`. 你将这个文件拷贝到你的应用中，并将它改名为 .java 的文件，并修改persistence.xml：

```
<property
    name="toplink.target-database"
    value="oracle.toplink.essentials.platform.database.H2Platform"/>
```

旧版本的 Glassfish 的属性名为 `toplink.platform.class.name`。

在 Glassfish 中使用 H2,拷贝 h2*.jar 到 `glassfish/glassfish/lib` 目录

使用 EclipseLink

在 EclipseLink 使用 H2，可以通过类 `org.eclipse.persistence.platform.database.H2Platform`。如果你使用的 EclipseLink 版本不支持，可以使用 OraclePlatform 替代，具体看 H2Platform

使用 Apache ActiveMQ

当使用 H2 作为 Apache ActiveMQ 的后端数据库,请使用 `TransactDatabaseLocker` 而不是默认的锁机制。否则,数据库文件将没有界限的增长。原因是默认锁机制是使用一个未提交的 `UPDATE` 事务,使得事务日志不会收缩（造成数据库文件增长)。`TransactDatabaseLocker` 使用 `SELECT ... FOR UPDATE` 而不是使用一个 `UPDATE` 语句,来解决问题。使用它,改变 ApacheMQ 配置元素 `<jdbcPersistenceAdapter>` 元素,属性 `databaseLocker ="org.apache.activemq.store.jdbc.adapter.TransactDatabaseLocker"`。然而,使用 MVCC 模式将再次导致同样的问题。因此,在这种情况下,请不要使用 MVCC 模式。另一个(更危险)解决方案是 `useDatabaseLock` 设置为 false。

在 NetBeans 中使用 H2

项目 H2 Database Engine Support For NetBeans 允许您在 IDE 中启动和停止服务器 H2。

有一个已知问题当使用 Netbeans SQL Execution Window:在执行查询之前,另一个查询 `SELECT COUNT(*) FROM <query>` 运行。这是一个查询的问题会修改状态,如 `SELECT SEQ.NEXTVAL`。在这种情况下,两个序列值分配而不是一个。

在 jOOQ 中使用 H2

jOOQ 添加一个 JDBC 薄层,允许 SQL 类型安全的建设,包括高级 SQL,存储过程和高级的数据类型。jOOQ 需要数据库模式作为基础代码生成。如果这是你的示例模式:

```
CREATE TABLE USER (ID INT, NAME VARCHAR(50));
```

使用命令行来运行 jOOQ 代码生成器:

```
java -cp jooq.jar;jooq-meta.jar;jooq-codegen.jar;h2-1.3.158.jar;.org.jooq.util.GenerationTool /codegen.xml
```

当 codegen.xml 在 classpath 并且包含了如下信息

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configuration xmlns="http://www.jooq.org/xsd/jooq-codegen-2.3.0.xsd">
    <jdbc>
        <driver>org.h2.Driver</driver>
        <url>jdbc:h2:~/test</url>
        <user>sa</user>
        <password></password>
    </jdbc>
    <generator>
        <name>org.jooq.util.DefaultGenerator</name>
        <database>
            <name>org.jooq.util.h2.H2Database</name>
            <includes>.*</includes>
            <excludes></excludes>
            <inputSchema>PUBLIC</inputSchema>
        </database>
        <generate></generate>
        <target>
            <packageName>org.jooq.h2.generated</packageName>
            <directory>./src</directory>
        </target>
    </generator>
</configuration>
```

使用生成的原件，可以执行如下查询s:

```
Factory create = new H2Factory(connection);
Result<UserRecord> result =
create.selectFrom(USER)
    .where(NAME.like("Johnny%"))
    .orderBy(ID)
    .fetch();
```

细节详见 jOOQ 主页 和 jOOQ Tutorial

在 Web 应用中使用 H2 数据库

在 Web 应用中使用数据库，可以有多种方式，这里有一些针对 Tomcat 和 JBoss 的例子。

内嵌模式

最简单（目前）的方法就是将数据库内嵌到应用中，这样就意味着应用启动的时候就打开了一个连接（好的办法是使用 Servlet 监听器，看下面的说明）。数据库能被多个 session 和应用访问，他们跟应用运行在一个进程内，大部分的 Servlet 容器只适用一个进程（如Tomcat），这些容器都是没有问题的（除非你使用集群）。Tomcat 使用多线程和多类加载器。如果多个应用同时访问同一个数据库，你需要将数据库的 jar 文件放在 `shared/lib` 或是 `server/lib` 目录。好的方案是 Web 应用启动时打开数据库，Web 应用停止时关闭数据库。如果是多个应用，只需要一个应用来处理启动和关闭。好的方案是一个 Session 一个连接，或者是一个请求（action）一个连接，连接使用完后尽可能的关闭它，当然不关闭并不会引起可怕的后果。

服务器模式

服务器模式是差不多的，但是它可以运行在其他的进程中。

使用 Servlet 监听去启动和停止数据库

增加 `h2*.jar` 文件到你的应用中，将下面的配置增加到你的 web.xml 中（在 `filter` 节下面的 `context-param`）：

```
<listener>
    <listener-class>org.h2.server.web.DbStarter</listener-class>
```

```
</listener>
```

关于具体访问数据库的细节，你可以看 DbStarter.java。在这个工具中缺省打开的内嵌数据库 URL 为`jdbc:h2:~/test`， 用户名 sa，密码 sa。如果你要去使用这个连接，你可以使用下面的访问方式：

```
Connection conn = getServletContext().getAttribute("connection");
```

DbStarter 也能够启动 TCP 服务，但是缺省状态下是不允许的。可以通过修改 web.xml 下的参数 `db.tcpServer` 来启用。下面是完整的配置选项，这些选项需要放在`description`标签和 `listener`/`filter` 标签中间：

```
<context-param>
    <param-name>db.url</param-name>
    <param-value>jdbc:h2:~/test</param-value>
</context-param>
<context-param>
    <param-name>db.user</param-name>
    <param-value>sa</param-value>
</context-param>
<context-param>
    <param-name>db.password</param-name>
    <param-value>sa</param-value>
</context-param>
<context-param>
    <param-name>db.tcpServer</param-name>
    <param-value>-tcpAllowOthers</param-value>
</context-param>
```

当 web 应用停止时，数据库连接将被自动关闭，如果通过 DbStarter 还启动了 TCP 服务，TCP 服务也将被自动关闭。

使用 **H2** 控制台 **Servlet**

H2 控制台是一个包含在 web 服务中的独立的应用，但是它也能作为一个servlet使用。为了做到这点，你需要将`h2*.jar`文件添加到你的应用中，在你的 web.xml 文件中增加下面的配置：

```
<servlet>
    <servlet-name>H2Console</servlet-name>
    <servlet-class>org.h2.server.web.WebServlet</servlet-class>
    <!--
    <init-param>
        <param-name>webAllowOthers</param-name>
        <param-value></param-value>
    </init-param>
    <init-param>
        <param-name>trace</param-name>
        <param-value></param-value>
    </init-param>
    -->
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>H2Console</servlet-name>
    <url-pattern>/console/*</url-pattern>
</servlet-mapping>
```

关于更多的细节，请参考`src/tools/WEB-INF/web.xml`

要创建一个合适的 H2 控制台的 web 应用，运行下面的命令：

```
build warConsole
```

Android

Android 设备(使用Dalvik VM)上可以使用这个数据库代替 SQLite。到目前为止,只有很少的测试和基准测试运行,但似乎性能类似于 SQLite,除了打开和关闭数据库没有优化 (H2大约需要0.2秒,和SQLite约0.02秒)。读操作似乎比 SQLite 快,而写操作慢点。到目前为止,只有很少的测试运行,一切似乎正常工作。全文搜索还没有测试,然而本地的全文搜索应该能工作。

使用 H 2而不是 SQLite 原因是:

- 全 Unicode 支持包括 UPPER() 和 LOWER()
- 流 API 用于 BLOB 和 CLOB 数据
- 全文搜索
- 多连接
- 用于定义方法和触发器
- 数据库文件加密
- 读写 CSV 文件 (该功能也可以在数据库外部使用)
- 引用完整性和检查约束
- 更好的数据类型和 SQL 支持
- 内存数据库、只读数据库、表关联
- 与其他数据库更好的兼容,简化应用程序移植
- 可能更好的性能（对读操作）
- 服务器模式(在不同的机器上通过 TCP/IP 访问同一个数据库)。

目前只支持 JDBC API (计划在将来的版本中支持 Android 数据库API)。常规的 H2 jar文件和小 `h2small-*.jar` 可以使用。为了创建更小的 jar 文件,运行命令 `./build.sh jarSmall` (Linux / Mac OS) 或 `build.bat jarSmall` (Windows)

数据库文件需要存储在一个可以被一个应用程序访问的地方。例子:

```
String url = "jdbc:h2:/data/data/" +
    "com.example.hello" +
    "/data/hello" +
    ";FILE_LOCK=FS" +
    ";PAGE_SIZE=1024" +
    ";CACHE_SIZE=8192";
Class.forName("org.h2.Driver");
conn = DriverManager.getConnection(url);
...
```

限制:使用连接池目前不支持,因为所需的javax.sql 类是不在 Android 上的。

CSV (Comma Separated Values) 支持

CSV（逗号分隔文件）文件在数据库系统中支持 `CSVREAD` 和 `CSVWRITE` 方法，也可以把它作为数据库之外的一个工具来使用。将数据库查询结果写成CSV文件

通过数据库读取 CSV 文件

使用 `CSVREAD`， 例子：

```
SELECT * FROM CSVREAD('test.csv');
```

请注意由于性能原因, `CSVREAD` 不应使用在 jion 内部。相反,先导入数据(可能到一个临时表),如果有必要创建所需的索引,然后查询这个表。

通过 CSV 文件导入数据

从CSV文件快速加载或导入数据(有时称为"批量加载")的方法是结合表创建和导入。可选地,该列名称和数据类型可以在创建表时设置。另一个选项 `INSERT INTO ... SELECT`。

```
CREATE TABLE TEST AS SELECT * FROM CSVREAD('test.csv');
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255))
    AS SELECT * FROM CSVREAD('test.csv');
```

通过数据库写 CSV 文件

使用 `CSVWRITE`,

```
CREATE TABLE TEST(ID INT, NAME VARCHAR);
INSERT INTO TEST VALUES(1, 'Hello'), (2, 'World');
CALL CSVWRITE('test.csv', 'SELECT * FROM TEST');
```

通过 Java 应用写 CSV 文件

使用 Csv 工具而无需数据库,

```
import java.sql.*;
import org.h2.tools.Csv;
import org.h2.tools.SimpleResultSet;
public class TestCsv {
    public static void main(String[] args) throws Exception {
        SimpleResultSet rs = new SimpleResultSet();
        rs.addColumn("NAME", Types.VARCHAR, 255, 0);
        rs.addColumn("EMAIL", Types.VARCHAR, 255, 0);
        rs.addRow("Bob Meier", "bob.meier@abcde.abc");
        rs.addRow("John Jones", "john.jones@abcde.abc");
        new Csv().write("data/test.csv", rs, null);
    }
}
```

通过 Java 应用读 CSV 文件

读 CSV 文件可以无需打开数据库,

```
import java.sql.*;
import org.h2.tools.Csv;
public class TestCsv {
    public static void main(String[] args) throws Exception {
        ResultSet rs = new Csv().read("data/test.csv", null, null);
        ResultSetMetaData meta = rs.getMetaData();
        while (rs.next()) {
            for (int i = 0; i < meta.getColumnCount(); i++) {
                System.out.println(
                    meta.getColumnLabel(i + 1) + ": " +
                    rs.getString(i + 1));
            }
            System.out.println();
        }
        rs.close();
    }
}
```

升级、备份和恢复

数据库升级

数据库升级的推荐方案是，老版本的数据库的数据备份成 SQL 脚本的方式，在新版本的数据库上执行这些 SQL 来恢复数据。

使用 **Script** 工具备份数据

备份数据库有多种方式。如可以直接拷贝数据库文件，但是不建议在数据库在使用的时候去拷贝文件，另外数据库文件是二进制的，不能直接读懂，并且数据库文件可能会比较大，推荐的备份方式是创建压缩的 SQL 脚本文件，并且 H2 提供了数据导出的 `Script` 工具：

```
java org.h2.tools.Script -url jdbc:h2:~/test -user sa -script test.zip -options compression zip
```

也可能通过 SQL 命令`SCRIPT`去备份数据库，关于更多的命令选项，请查看 SQL 命令`SCRIPT`。备份也能通过远程来做，但是文件被创建在服务器上，你可以通过 FTP 服务获取备份的脚本文件。

用脚本恢复数据

从一个SQL脚本文件恢复数据库，你可以使用`RunScript`工具：

```
java org.h2.tools.RunScript -url jdbc:h2:~/test -user sa -script test.zip -options compression zip
```

关于更多的命令的选项，请参考 SQL 命令 `RUNSCRIPT`。恢复也能通过远程来实现，但是恢复的文件需要在服务器上。可以通过FTP服务器上传恢复需要的脚本文件。也可以通过`RUNSCRIPT`执行 SQL 脚本，SQL 脚本文件内也可以引用另外的 SQL 脚本文件，在服务器模式下，也可以远程执行 SQL 脚本，但是要求脚本文件和被引用的脚本未见都在服务器上。

在线备份

`BACKUP` SQL 语句和 `Backup` 工具都能创建全库的备份文件的压缩 zip 包。但是，这个文件的内容并不可读

相比脚本方式，BACKUP命令并不锁定数据库对象，也不阻塞用户，但是 BACKUP 命令备份结果是事务一致的。：

```
BACKUP TO 'backup.zip'
```

`Backup`工具 (org.h2.tools.Backup)不能创建在线备份；程序在执行的时候，数据库不能使用。

并不支持数据库运行的同时创建数据库备份，除非是文件系统支持创建快照，但是快照不能保证数据拷贝顺序的正确性。

命令行工具

H2数据库提供了一组命令行工具，如果你需要了解这些工具，使用参数 `-?`，如：

```
java -cp h2*.jar org.h2.tools.Backup -?
```

命令行工具有：

- Backup 创建数据库备份

- ChangeFileEncryption 允许改变文件加密密码和数据库的加密算法
- Console 启动基于浏览器的 H2 控制台
- ConvertTraceFile 转换 `.trace.db` 文件到 Java 应用和 SQL 脚本
- CreateCluster 从一个独立的数据库服务创建集群
- DeleteDbFiles 删除所有的数据库文件
- Recover 恢复损坏的数据库
- Restore 从数据库备份中恢复数据库
- RunScript 运行数据库 SQL 脚本
- Script 为数据库备份或迁移导出 SQL 脚本
- Server 启动 H2 服务模式
- Shell 命令行工具

这些工具也能在程序中通过调用相应的方法来使用，相关详细的调用说明，请参考 JavaDoc 文档。

Shell 工具

Shell 工具是最简单的命令行工具，开始时，输入:

```
java -cp h2*.jar org.h2.tools.Shell
```

你会要求输入数据库URL、JDBC 驱动、用户名和密码。连接设置也可以作为命令行参数设置。连接后,你会得到的列表选项。内置命令不需要以分号结束,但只是执行 SQL 语句,需要以分号 `;` 作为结束。这允许输入多行语句:

```
sql> select * from test
...> where id = 0;
```

默认情况下,结果是打印成表。结果有许多列,考虑使用模式列表:

```
sql> list
Result list mode is now on
sql> select * from test;
ID  : 1
NAME: Hello

ID  : 2
NAME: World
(2 rows, 0 ms)
```

使用 OpenOffice 基础框架

OpenOffice.org 基础框架支持通过 JDBC 连接数据库。你也可以通过 OpenOffice 框架连接到 H2 数据库。首先将 JDBC 驱动增加到 OpenOffice 中，下面的步骤可以连接到H2数据库：

- 启动 OpenOffice Writer，进入[Tools], [Options]
- 确认你在 OpenOffice.org/Java 中选择了 JAVA 运行环境
- 单击 [Class Path...], [Add Archive...]
- 选择你的 h2 的 jar 文件（本机上的路径可以由你选择）
- 单击 [OK] (按要求的点击), 停止OpenOffice (包括 Quickstarter)
- 启动 OpenOffice 框架
- 连接到一个存在的数据库；选择 [JDBC]; [Next]
- 输入数据库URL，如： `jdbc:h2:~/test`

- 输入 JDBC 驱动类： `org.h2.Driver`

你可以访问存于用户当前目录的数据库。

使用 H2 数据通过 NeoOffice (去掉 X11 的 OpenOffice)：

- 在 NeoOffice，到[NeoOffice]， [Preferences]
- 在[NeoOffice]页下找到[Java]
- 单击[Class Path], [Add Archive...]
- 选择 h2 的 jar 文件（本地目录，任你选择合适的目录）
- 单击[OK]（根据需要），重启 NeoOffice

现在，你可以通过"Database Wizard"创建一新的数据库：

- 单击[File], [New], [Database]
- 选择[Connect to existing database]，并且选择[JDBC]，单击next
- 输入数据源的URL，例如：`jdbc:h2:~/test`
- JDBC驱动类： org.h2.Driver

其他的在 NeoOffice 中使用 H2 的方法：

- 将 H2 的 jar 包打包到一个扩展包中
- 在 NeoOffice 中作为扩展 Java 包进行安装

这个能通过使用 NetBeans OpenOffice 插件来创建。详细看Extensions Development

Java Web Start / JNLP

当使用 Java Web Start / JNLP（JAVA网络加载协议），允许访问标签必须被设置在 .jnlp 文件，并且 .jar 的应用文件必须被签名，否则，当你试着去写文件系统，下面的异常将会被抛出：`java.security.AccessControlException`：拒绝访问(`java.io.FilePermission ... read`). 如访问标签：

```
<security>
    <all-permissions/>
</security>
```

使用连接池

如果 H2 的数据库已经打开，打开一个连接很快。如果要打开和关闭许多连接，使用连接池，可以提升性能。H2 包含了一个简单的连接池，它是基于Christian d'Heureuse 的 Mini Connection Pool Manager 。还有其他更复杂的开源连接池可以使用，如Apache Commons DBCP。H2 从内置连接池获取连接比使用 `DriverManager.getConnection()` 快两倍左右。内置的连接池使用方法如下：

```
import java.sql.*;
import org.h2.jdbcx.JdbcConnectionPool;
public class Test {
    public static void main(String[] args) throws Exception {
        JdbcConnectionPool cp = JdbcConnectionPool.create(
            "jdbc:h2:~/test", "sa", "sa");
        for (int i = 0; i < args.length; i++) {
            Connection conn = cp.getConnection();
            conn.createStatement().execute(args[i]);
            conn.close();
        }
        cp.dispose();
```

```
    }
}
```

全文检索

H2 包含了两种全文检索的实现。一种通过使用 Apache Lucene 来实现，另一种是通过存储索引文件到数据库里的一张特殊表来实现的（私有实现）。

使用私有全文检索

通过下面的调用来实现初始化：

```
CREATE ALIAS IF NOT EXISTS FT_INIT FOR "org.h2.fulltext.FullText.init";
CALL FT_INIT();
```

如果你要用到私有全文检索，你需要在每个数据库里都初始化它。然后你能创建一张表用于全文检索的索引，如：

```
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR);
INSERT INTO TEST VALUES(1, 'Hello World');
CALL FT_CREATE_INDEX('PUBLIC', 'TEST', NULL);
```

PUBLIC 是 schema 的名字， TEST 是索引表名。字段名列表是可选的，在上面的例子中，所有的字段都被索引，索引的更新是实时的，使用下面的查询语句可以进行搜索：

```
SELECT * FROM FT_SEARCH('Hello', 0, 0);
```

下面的语句将得到一个指定内容的搜索结果集。

```
QUERY: "PUBLIC"."TEST" WHERE "ID"=1
```

删除表上的索引

```
CALL FT_DROP_INDEX('PUBLIC', 'TEST');
```

要得到原始的数据，需要使用 `FT_SEARCH_DATA('Hello', 0, 0);` 。结果包含字段 SCHEMA (schema名), TABLE (表名), COLUMNS (字段名数组), 和 KEYS (对象数组)。可以和表做连接，使用连接如下： `SELECT T.* FROM FT_SEARCH_DATA('Hello', 0, 0) FT, TEST T WHERE FT.TABLE='TEST' AND T.ID=FT.KEYS[0];`

你也能在 Java 应用中使用索引：

```
org.h2.fulltext.FullText.search(conn, text, limit, offset);
org.h2.fulltext.FullText.searchData(conn, text, limit, offset);
```

用 **Lucene** 实现的全文检索

使用 Lucene 实现的全文检索，你需要将 Lucene 加入到 classpath 中，并且处理相关的依赖，如果你使用 H2 控制台，你能增加 Lucene 的 jar 文件到环境变量 `H2DRIVERS` 或 `CLASSPATH` 中。通过下面的语句初始化 Lucene 全文检索：

```
CREATE ALIAS IF NOT EXISTS FTL_INIT FOR "org.h2.fulltext.FullTextLucene.init";
CALL FTL_INIT();
```

如果你需要是使用 Lucene 全文检索，你每个数据库都需要初始化。你也能通过下面的语句创建全文索引表：

```
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR);
INSERT INTO TEST VALUES(1, 'Hello World');
CALL FTL_CREATE_INDEX('PUBLIC', 'TEST', NULL);
```

PUBLIC 是 schema 名, TEST 是表名. 字段名列表是可选的，在上面的例子中，所有的字段都被索引，索引的更新是实时的，使用下面的查询语句可以进行搜索：

```
SELECT * FROM FTL_SEARCH('Hello', 0, 0);
```

下面的语句将得到一个指定内容的搜索结果集：

```
QUERY: "PUBLIC"."TEST" WHERE "ID"=1
```

表上的索引(警告称,这将对所有的整个数据库的全文索引进行检索):

```
CALL FTL_DROP_INDEX('PUBLIC', 'TEST');
```

要得到原始的数据，需要使用 `FTL_SEARCH_DATA('Hello', 0, 0);`。结果包含字段 SCHEMA (schema名), TABLE (表名), COLUMNS (字段名数组), 和KEYS (对象数组)。可以和表做连接，使用连接如下 `SELECT T.* FROM FTL_SEARCH_DATA('Hello', 0, 0) FT, TEST T WHERE FT.TABLE='TEST' AND T.ID=FT.KEYS[0]`;

你也能在JAVA应用中使用索引：

```
org.h2.fulltext.FullTextLucene.search(conn, text, limit, offset);
org.h2.fulltext.FullTextLucene.searchData(conn, text, limit, offset);
```

Lucene 搜索支持全文搜索仅在特定的列。列名必须大写(如果原始列双援引除外)。为列名下划线(_),另一个强调需要添加。例子:

```
CREATE ALIAS IF NOT EXISTS FTL_INIT FOR "org.h2.fulltext.FullTextLucene.init";
CALL FTL_INIT();
DROP TABLE IF EXISTS TEST;
CREATE TABLE TEST(ID INT PRIMARY KEY, FIRST_NAME VARCHAR, LAST_NAME VARCHAR);
CALL FTL_CREATE_INDEX('PUBLIC', 'TEST', NULL);
INSERT INTO TEST VALUES(1, 'John', 'Wayne');
INSERT INTO TEST VALUES(2, 'Elton', 'John');
SELECT * FROM FTL_SEARCH_DATA('John', 0, 0);
SELECT * FROM FTL_SEARCH_DATA('LAST_NAME:John', 0, 0);
CALL FTL_DROP_ALL();
```

Lucene 全文搜索实现内部并不是同步的。如果你同时更新数据库和查询全文搜索(直接使用 Java API 的 H2 或 Lucene 本身),您需要确保操作正确同步。如果不是这样的话,你可能会有异常 `org.apache.lucene.store.AlreadyClosedException: this IndexReader is closed`。

用户定义变量

数据库支持用户自定义变量，自定义变量使用 @ 开头，能够被用于任何表达式和参数中。变量是不能持久的，作为范围为 session，这就意味着变量只在定义它的那个 session 里是有效的。一个变量通常使用 SET命 令来声明：

```
SET @USER = 'Joe';
```

变量也可以通过使用 SET() 方法来改变值。在查询中可以直接使用：

```
SET @TOTAL = NULL;
SELECT X, SET(@TOTAL, IFNULL(@TOTAL, 1.) * X) F FROM SYSTEM_RANGE(1, 50);
```

变量不能被设置为 NULL 值，变量的类型是变量自动分配的，也就是说，在变量使用前变量的定义并不是必须（也不是必需）的，在声明变量时没有限制，大对象（LOB）也被支持。

日期和时间

日期、时间、时间戳的值支持ISO 8601格式，格式还包含了时区：

```
CALL TIMESTAMP '2008-01-01 12:00:00+01:00';
```

如果未设置时区，将使用系统中当前的时区。日期和时间信息存储在 H2 数据库文件,不包含时区信息。如果数据库打开使用另一个系统时区,日期和时间将是相同的。这意味着如果你存储的值“2000-01-01 12:00:00”在一个时区,然后再关闭数据库并打开数据库在不同的时区,你还将得到“2000-01-01 12:00:00”。请注意,更改时区 H2 加载驱动程序不支持。

使用 Spring

使用 TCP 服务器

在 Spring 配置如下，来启动和关闭 H2 TCP 服务器

```
<bean id = "org.h2.tools.Server"
            class="org.h2.tools.Server"
            factory-method="createTcpServer"
            init-method="start"
            destroy-method="stop">
    <constructor-arg value="-tcp,-tcpAllowOthers,-tcpPort,8043" />
</bean>
```

destroy-method 方法能防止热重部署和重启异常。

错误代码不兼容

有一个不相容，就是 Spring JdbcTemplate 和 H2 version 1.3.154版本及更新的版本,因为错误代码有变化。这将导致 JdbcTemplate 不是检测重复的关键条件,所以一个 DataIntegrityViolationException 代替 DuplicateKeyException 抛出。参见 SPR-8235 的问题。解决方法是添加以下 XML 文件的根路径:

```
<beans
    xmlns="http://www.springframework.org/schema/beans"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
        "http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd"
    >
    <import resource="classpath:org/springframework/jdbc/support/sql-error-codes.xml"/>
    <bean id = "H2" class="org.springframework.jdbc.support.SQLErrorCodes">
        <property name="badSqlGrammarCodes">
            <value>
                42000,42001,42101,42102,42111,42112,42121,42122,42132
            </value>
        </property>
        <property name="duplicateKeyCodes">
            <value>23001,23505</value>
        </property>
        <property name="dataIntegrityViolationCodes">
            <value>22003,22012,22025,23000</value>
        </property>
        <property name="dataAccessResourceFailureCodes">
            <value>90046,90100,90117,90121,90126</value>
        </property>
        <property name="cannotAcquireLockCodes">
            <value>50200</value>
        </property>
    </bean>
</beans>
```

OSGi

标准的 H2 jar 可以下降为一捆在一个 OSGi 容器中。H2 实现了 OSGi Service Platform Release 4 Version 4.2 Enterprise Specification 中定义 JDBC 服务 。H2 数据源工厂服务注册以下属性: `OSGI_JDBC_DRIVER_CLASS=org.h2.Driver` 和 `OSGI_JDBC_DRIVER_NAME=H2` 。 `OSGI_JDBC_DRIVER_VERSION` 属性反映了版本的驱动程序。

支持以下标准配置属性: `JDBC_USER, JDBC_PASSWORD, JDBC_DESCRIPTION, JDBC_DATASOURCE_NAME, JDBC_NETWORK_PROTOCOL, JDBC_URL, JDBC_SERVER_NAME, JDBC_PORT_NUMBER` 。任何其他标准属性将被拒绝。非标准的属性将被传递给 H2 连接的 URL

Java Management Extension (JMX)

支持 JMX 管理, 但不是默认启用。为了启用 JMX,追加 `;JMX=TRUE` 到数据库 URL， 当数据库打开时。各种工具支持 JMX,其中一个是 `jconsole` 工具。当打开 `jconsole` ,可以连接到打开的数据库的进程(使用服务器模式时,您需要连接到服务器进程)。然后去MBeans 部分。 在 `org.h2` 下每个数据库都会发现一个实体。这个实体对象名称的是数据库短名称,加上路径(每个冒号被替换为下划线字符)。

以下属性和操作的支持:

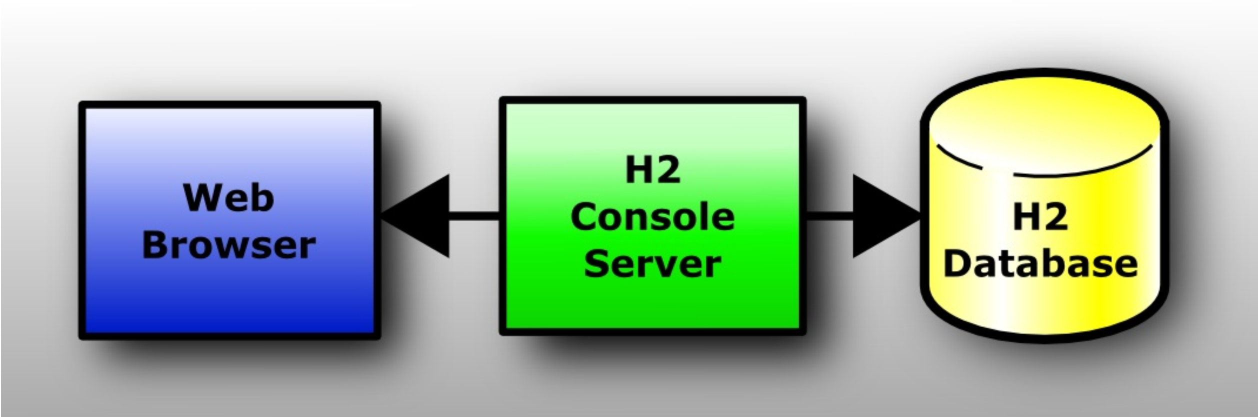
- CacheSize:当前使用的缓存大小， 单位KB
- CacheSizeMax(读/写):最大缓存大小， 单位KB
- Exclusive:这个数据库是否在独占模式
- FileReadCount:数据库打开后，文件读取操作的数量
- FileSize:文件大小， 单位KB
- FileWriteCount:数据库打开后，文件写操作的数量
- FileWriteCountTotal:数据库创建后，文件写操作的数量
- LogMode(读/写):当前事务日志模式。有关详细信息,请参阅 `SET LOG`
- Mode:兼容性模式(`REGULAR` 如果没有使用兼容模式)
- MultiThreaded:如果启用了多线程就是 true
- Mvcc:如果启用了Mvcc 就是 true
- ReadOnly:如果数据库是只读的就是 true
- TraceLevel (读/写):文件跟踪级别
- Version:使用的数据库版本

- listSettings:列出数据库设置
- listSessions:开放会话列表,包括当前执行语句(如果有的话)和锁定的表(如果有的话)

要启用 JMX,可能需要设置 JVM 系统属性 `com.sun.management.jmxremote` 和 `com.sun.management.jmxremote.port`

使用和启动H2管理系统

H2 管理系统让你能够通过一个浏览器对 H2 的 SQL 数据库进行管理操作。H2 管理系统不仅可以连接 H2 数据库，也可以连接其他支持 JDBC API 的数据库。



这是一个 C/S 应用，在服务器和客户端（浏览器）上都要运行 H2 的管理程序。根据平台不同，H2 管理系统支持多种启动应用的方式：

操作系统	启动
Windows	点击 [Start], [All Programs], [H2], 和 [H2 Console (Command Line)]。系统的系统托盘可以看到 H2 的图标。如果看不到可能是 Java 没有安装正确。浏览器访问http://localhost:8082。
Windows	打开文件浏览器，切换目录到h2/bin，双击运行h2.bat。一个控制台窗口将弹出，如果有问题，将有错误信息在这个窗口里显示。一个浏览器窗口将被打开，指向的URL是http://localhost:8082，是H2管理系统的登录页面。
Any	双击h2*.jar文件。前提是 .jar 文件能正确的被 Java 打开。
Any	开启控制台窗口，切换目录到h2/bin，执行命令：`java -cp h2*.jar org.h2.tools.Server`

防火墙

在你启动服务时，如果你安装了防护墙，你可能会收到一个防护墙的安全警告。如果不需要其他计算机访问你这台计算机上的H2数据库，你可以让防火墙阻塞H2对外服务的端口，但是本地计算机仍可以访问这些端口。当你需要其他计算机也能访问这台计算机的H2数据库时，你需要让防火墙开放H2对外服务的端口。

有报告显示使用 Kaspersky(卡巴斯基)7.0 的防火墙时，使用 IP 地址访问本地的 H2 时，速度非常的缓慢，替代的方案是使用'localhost'代替IP 地址来访问。

一个简单的防火墙已经集成到H2的服务器中，其他的计算机缺省状态下不能连接到服务器，如果需要其他计算机能连接到H2服务器，到'Preferences'（偏好），选择'Allow connections from other computers'（允许从其他计算连接）即可

JAVA测试

打开一个命令行窗口，输入下面的命令，检测JAVA的版本：

```
java -version
```

如果你得到错误的信息，你可能未安装 JDK，或是需要将 Java 的可执行文件路径加入到环境变量 PATH 中。

错误信息'Port may be in use'(端口被占用)

你可能在启动一个 H2 控制台实例时，出现错误信息"The Web server could not be started. Possible cause: another server is already running...".（WEB服务器不能启动，可能的原因：另外一个服务器已经在运行了）。使用不同的端口，可以在一台计算机上启动多个控制台程序， 但是一般都不会做。

使用其他端口

如果端口已经被其他应用占用，你需要使用其他端口来启动 H2 控制台。改变 H2 的控制台端口需要修改配置文件 `.h2.server.properties` 。这个文件存储在用户目录下(在 Windows 系统中，这个文件通常在 `Documents and Settings/<username>`)。这个相应的入口实体是 webPort.

使用浏览器连接到服务器

服务器启动成功后，你就可以使用 WEB 浏览器访问服务，浏览器需要支持JavaScript。在启动的服务器上启动浏览器，打开 URL http://localhost:8082。在启动服务器之外的计算机上，你需要提供启动服务器的IP地址，如http://192.168.0.2:8082.如果你在服务器上启用了SSL， URL需要使用https://开头.

多个并发会话

支持多个并发的浏览器会话。由于数据对象是存储在服务器上的，同时工作的会话数受限于服务器的内存。

登录

在登录页，你提交连接信息就可以登录到数据库。设置 JDBC 作为连接数据库的驱动，填入JDBC URL、用户名、密码，单击[Connect]。你能保存和恢复以前设置的信息，这些设置都存储在属性文件中。

错误信息

错误信息用红色标识，你能通过单击消息显示或隐藏异常的堆栈信息。

附加数据库驱动

通过增加 jar 的本地驱动文件到环境变量来附加数据库驱动 (MySQL, PostgreSQL, HSQLDB,...)。环境变量包括 H2DRIVERS 和 CLASSPATH，以 Windows 为例：如要增加数据库驱动 `C:\Programs\hsqldb\lib\hsqldb.jar`，设置环境变量 H2DRIVERS 为 `C:\Programs\hsqldb\lib\hsqldb.jar`

多个驱动可以被设置，每个驱动之间通过 `;` 分号分隔（Windows），其他系统通过 `:` 冒号分隔。在路径中空格被支持，但是这些设置不能被引用。

使用 **H2** 控制台

H2 控制台主页面分为三个主要的部分：顶部的工具栏，左边的是对象树，右边的是查询和结果输出栏。数据库对象（如表）都被列在左边的树形上。在查询栏上输入SQL语句点击 [Run]，结果就被输出到命令行的下面。

增加表名和字段名

可以通过在树上点击增加表名和字段名到脚本。如果单击表，当这个查询栏是空的时候， `SELECT * FROM...` 将被自动增加到查询栏。当输入一个表的查询时，对象树上将自动的展开这张表。例如，你输入 `SELECT * FROM TEST T WHERE T`，对象树上的表 TEST 将自动的展开

断开连接和停止应用

断开数据库，点击工具栏上的 [Disconnect] 即可，这个时候，数据库服务仍在继续运行，等待着一个新的会话进行连接。

停止服务需要右键点击系统托盘的 H2 图标，选择 [Exit]。如果没有系统托盘的 H2 图标，切换到 [Preferences] 单击 [Shutdown]，在Windows 上在服务器启动的窗口下按[Ctrl]+[C]，或者直接关闭 Windows 上的控制台窗口。

H2 控制台的特殊语法

H2 控制台支持几个内置命令。这些都在 H2 控制台做解析,所以他们使用任何数据库。内置命令需要在声明的开始(在任何备注之前),否则他们是无法正确解析。如果有疑问,之前添加 `;` 命令。

Command(s)	Description
@autocommit_true; @autocommit_false;	Enable or disable autocommit.
@cancel;	Cancel the currently running statement.
@columns null null TEST; @index_info null null TEST; @tables; @tables null null TEST;	Call the corresponding `DatabaseMetaData.get` method. Patterns are case sensitive (usually identifiers are uppercase). For information about the parameters, see the Javadoc documentation. Missing parameters at the end of the line are set to null. The complete list of metadata commands is: `@attributes, @best_row_identifier, @catalogs, @columns, @column_privileges, @cross_references, @exported_keys, @imported_keys, @index_info, @primary_keys, @procedures, @procedure_columns, @schemas, @super_tables, @super_types, @tables, @table_privileges, @table_types, @type_info, @udts, @version_columns`
@edit select * from test;	Use an updatable result set.
@generated insert into test() values();	Show the result of `Statement.getGeneratedKeys()` .
@history;	List the command history.
@info;	Display the result of various `Connection` and `DatabaseMetaData` methods.
@list select * from test;	Show the result set in list format (each column on its own line, with row numbers).
@loop 1000 select ?, ?/*rnd*/; @loop 1000 @statement select ?;	Run the statement this many times. Parameters (`?`) are set using a loop from 0 up to x - 1. Random values are used for each `?/*rnd*/` . A Statement object is used instead of a PreparedStatement if `@statement` is used. Result sets are read until `ResultSet.next()` returns `false` . Timing information is printed.
@maxrows 20;	Set the maximum number of rows to display.
@memory;	Show the used and free memory. This will call `System.gc()` .
@meta select 1;	List the `ResultSetMetaData` after running the query.
@parameter_meta select ?;	Show the result of the `PreparedStatement.getParameterMetaData()` calls. The statement is not executed.
@prof_start; call hash('SHA256', '', 1000000); @prof_stop;	Start/stop the built-in profiling tool. The top 3 stack traces of the statement(s) between start and stop are listed (if there are 3).
@prof_start; @sleep 10; @prof_stop;	Sleep for a number of seconds. Used to profile a long running query or operation that is running in another session (but in the same process).
@transaction_isolation; @transaction_isolation 2;	Display (without parameters) or change (with parameters 1, 2, 4, 8) the transaction isolation level.

设置 H2 控制台

H2 控制台的设置信息存储在配置文件 `.h2.server.properties`，该文件存放在你的用户目录下。在 Windows 上，用户目录通

常是 `C:\Documents and Settings\[username]` 或者 `C:\Users\[username]`。H2 控制台第一次启动时将自动创建应用所需要包含的配置文件。 支持的配置如下：

- webAllowOthers: 允许其他电脑连接
- webPort: H2 控制台的端口
- webSSL: 使用加密的 TLS (HTTPS) 连接 * 除了这些设置,列出了最近使用连接的属性的格式 <number>=<name>|<driver>|<url>|<user> 使用转义字符 `\`。举例 `1=Generic H2 (Embedded)|org.h2.Driver|jdbc\:h2\:~/test|sa`

使用 JDBC 连接数据库

JAVA应用要连接到数据库，首先需要加载数据库驱动，然后获得一个数据库连接，下面是一个简单的例子：

```
import java.sql.*;
public class Test {
    public static void main(String[] a)
            throws Exception {
        Class.forName("org.h2.Driver");
        Connection conn = DriverManager.
            getConnection("jdbc:h2:~/test", "sa", "");
        // add application code here
        conn.close();
    }
}
```

代码中通过 `Class.forName(...)` 来加载驱动，通过 `DriverManager.getConnection()` 来打开一个连接，驱动名为"org.h2.Driver"。数据库 URL 总是使用 `jdbc:h2:` 来标识， `getConnection()` 的第二个参数是用户名（ `sa` 作为系统超级管理员的一个例子），第三个参数是密码，用户名是不区分大小写，但是密码是大小写区分的。

创建新一个新数据库

缺省情况下，如果 URL 指定的数据库并不存在，一个新的空的数据库将被自动的创建。创建数据库的用户自动成为这个数据库的超级管理员。

自动创建新库也可以通过特殊的 URL 进行屏蔽，参见打开一个存在的数据库。

使用服务器模式

H2 目前支持三种服务器模式：web 服务器模式（H2 控制台）、TCP 服务器模式（C/S连接）和 PG 服务器模式（PostgreSQL 客户端）。可以通过多种方式启动服务器模式，通常的方式是通过 `Server` 工具。 开启服务器无需开启数据库——数据库会在客户端连接时开启

通过命令行启动 Server 工具

缺省设置下，输入下面命令并执行能启动 Server 工具：

```
java -cp h2*.jar org.h2.tools.Server
```

通过下面的命令行，可以查看服务器启动命令行的参数及缺省值：

```
java -cp h2*.jar org.h2.tools.Server -?
```

参数允许服务器工具启动到其他端口或者只是部分启动。

连接到 TCP 服务器

为了能远程通过 TCP 服务器访问数据库，使用下面的连接驱动和数据库URL：

• JDBC驱动类： `org.h2.Driver` • 数据库URL: `jdbc:h2:tcp://localhost/~/test`

关于数据库 URL，可以查看《特性》这章。注意不能通过浏览器访问这个 URL，只能通过 H2 客户端（通过 JDBC）。

在应用内部启动TCP服务

在JAVA应用内部，也可以通过代码来实现TCP服务的启动和停止，例子代码如下：

```
import org.h2.tools.Server;
...
// 启动 TCP Server
Server server = Server.createTcpServer(args).start();
...
// 关闭 TCP Server
server.stop();
```

从其他程序关闭 TCP 服务器

可以从另外的程序关闭 TCP 服务器，使用下面命令行：

```
java org.h2.tools.Server -tcpShutdown tcp://localhost:9092
```

在用户应用中关闭服务器，使用:

```
org.h2.tools.Server.shutdownTcpServer("tcp://localhost:9094");
```

这个功能将仅仅关闭 TCP 服务器。如果相同的程序有其他服务器，他们不会被关闭，而是继续执行。为了避免覆盖在数据库下次打开时，在调用这个方法时，所有的到数据库的连接将会关闭。要实现远程关闭服务器，需启用远程连接。关闭一个 TCP 服务器可以通过选项 `-tcpPassword` 来保护 (启动和关闭 TCP 服务器也要用这个密码)

使用 Hibernate

H2 数据库支持 Hibernate 3.1及以上的版本。 你能够使用 HSQLDB 方言，或是 H2 自己的方言。注意的是，在 Hibernate 中包含的 H2 方言有BUG，针对这些 BUG 的补丁已经被发布和修复，见https://hibernate.atlassian.net/browse/HHH-3401。你能够将它改名为H2Dialect.java，直接把它包含在你的应用中即可使用，或者使用 已经修复了这个问题的 Hibernate 的版本。

当使用 Hibernate,尝试使用 `H2Dialect` 如果可能的话。当使用 `H2Dialect` ,兼容性模式比如 `MODE=MySQL` 是不支持的。当使用兼容模式时,使用 Hibernate 相应的数据库的方言，而不是 `H2Dialect` ;但请注意 H2 不支持所有数据库的所有功能。

使用 TopLink 和 Glassfish

在 Glassfish （或 Sun AS）中使用 H2，设置 Datasource Classname 为 `org.h2.jdbcx.JdbcDataSource` 。可以通过图形界面进行设置[Application Server] - [Resources] - [JDBC] - [Connection Pools], 或者编辑文件 `sun-resources.xml` ：修改元素 `jdbc-connection-pool` ，设置属性 `datasource-classname` 为 `org.h2.jdbcx.JdbcDataSource` 。

H2 数据库是兼容 HSQLDB 和 PostgreSQL。如果要使用 H2 的特殊属性，需要使用 `H2Platform` ，源代码在 `src/tools/oracle/toplink/essentials/platform/database/DatabasePlatform.java.txt` . 你将这个文件拷贝到你的应用中，并将它改名为 .java 的文件，并修改persistence.xml：

```
<property
    name="toplink.target-database"
    value="oracle.toplink.essentials.platform.database.H2Platform"/>
```

旧版本的 Glassfish 的属性名为 `toplink.platform.class.name`。

在 Glassfish 中使用 H2,拷贝 h2*.jar 到 `glassfish/glassfish/lib` 目录

使用 EclipseLink

在 EclipseLink 使用 H2，可以通过类 `org.eclipse.persistence.platform.database.H2Platform`。如果你使用的 EclipseLink 版本不支持，可以使用 OraclePlatform 替代，具体看 H2Platform

使用 Apache ActiveMQ

当使用 H2 作为 Apache ActiveMQ 的后端数据库,请使用 `TransactDatabaseLocker` 而不是默认的锁机制。否则,数据库文件将没有界限的增长。原因是默认锁机制是使用一个未提交的 `UPDATE` 事务,使得事务日志不会收缩（造成数据库文件增长)。`TransactDatabaseLocker` 使用 `SELECT ... FOR UPDATE` 而不是使用一个 `UPDATE` 语句,来解决问题。使用它,改变 ApacheMQ 配置元素 `<jdbcPersistenceAdapter>` 元素,属性 `databaseLocker ="org.apache.activemq.store.jdbc.adapter.TransactDatabaseLocker"`。然而,使用 MVCC 模式将再次导致同样的问题。因此,在这种情况下,请不要使用 MVCC 模式。另一个(更危险)解决方案是 `useDatabaseLock` 设置为 false。

在 NetBeans 中使用 H2

项目 H2 Database Engine Support For NetBeans 允许您在 IDE 中启动和停止服务器 H2。

有一个已知问题当使用 Netbeans SQL Execution Window:在执行查询之前,另一个查询 `SELECT COUNT(*) FROM <query>` 运行。这是一个查询的问题会修改状态,如 `SELECT SEQ.NEXTVAL`。在这种情况下,两个序列值分配而不是一个。

在 jOOQ 中使用 H2

jOOQ 添加一个 JDBC 薄层,允许 SQL 类型安全的建设,包括高级 SQL,存储过程和高级的数据类型。jOOQ 需要数据库模式作为基础代码生成。如果这是你的示例模式:

```
CREATE TABLE USER (ID INT, NAME VARCHAR(50));
```

使用命令行来运行 jOOQ 代码生成器:

```
java -cp jooq.jar;jooq-meta.jar;jooq-codegen.jar;h2-1.3.158.jar;.org.jooq.util.GenerationTool /codegen.xml
```

当 codegen.xml 在 classpath 并且包含了如下信息

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configuration xmlns="http://www.jooq.org/xsd/jooq-codegen-2.3.0.xsd">
    <jdbc>
        <driver>org.h2.Driver</driver>
        <url>jdbc:h2:~/test</url>
        <user>sa</user>
        <password></password>
    </jdbc>
    <generator>
        <name>org.jooq.util.DefaultGenerator</name>
        <database>
            <name>org.jooq.util.h2.H2Database</name>
            <includes>.*</includes>
            <excludes></excludes>
```

```
            <inputSchema>PUBLIC</inputSchema>
        </database>
        <generate></generate>
        <target>
            <packageName>org.jooq.h2.generated</packageName>
            <directory>./src</directory>
        </target>
    </generator>
</configuration>
```

使用生成的原件，可以执行如下查询s:

```
Factory create = new H2Factory(connection);
Result<UserRecord> result =
create.selectFrom(USER)
    .where(NAME.like("Johnny%"))
    .orderBy(ID)
    .fetch();
```

细节详见 jOOQ 主页 和 jOOQ Tutorial

在 Web 应用中使用 H2 数据库

在 Web 应用中使用数据库，可以有多种方式，这里有一些针对 Tomcat 和 JBoss 的例子。

内嵌模式

最简单（目前）的方法就是将数据库内嵌到应用中，这样就意味着应用启动的时候就打开了一个连接（好的办法是使用 Servlet 监听器，看下面的说明）。数据库能被多个 session 和应用访问，他们跟应用运行在一个进程内，大部分的 Servlet 容器只适用一个进程（如Tomcat），这些容器都是没有问题的（除非你使用集群）。Tomcat 使用多线程和多类加载器。如果多个应用同时访问同一个数据库，你需要将数据库的 jar 文件放在 `shared/lib` 或是 `server/lib` 目录。好的方案是 Web 应用启动时打开数据库，Web 应用停止时关闭数据库。如果是多个应用，只需要一个应用来处理启动和关闭。好的方案是一个 Session 一个连接，或者是一个请求（action）一个连接，连接使用完后尽可能的关闭它，当然不关闭并不会引起可怕的后果。

服务器模式

服务器模式是差不多的，但是它可以运行在其他的进程中。

使用 Servlet 监听去启动和停止数据库

增加 `h2*.jar` 文件到你的应用中，将下面的配置增加到你的 web.xml 中（在 `filter` 节下面的 `context-param` ）：

```
<listener>
    <listener-class>org.h2.server.web.DbStarter</listener-class>
</listener>
```

关于具体访问数据库的细节，你可以看 DbStarter.java。在这个工具中缺省打开的内嵌数据库 URL 为 `jdbc:h2:~/test` ， 用户名 sa，密码 sa。如果你要去使用这个连接，你可以使用下面的访问方式：

```
Connection conn = getServletContext().getAttribute("connection");
```

DbStarter 也能够启动 TCP 服务，但是缺省状态下是不允许的。可以通过修改 web.xml 下的参数 `db.tcpServer` 来启用。下

面是完整的配置选项，这些选项需要放在`description`标签和 `listener` / `filter` 标签中间：

```
<context-param>
    <param-name>db.url</param-name>
    <param-value>jdbc:h2:~/test</param-value>
</context-param>
<context-param>
    <param-name>db.user</param-name>
    <param-value>sa</param-value>
</context-param>
<context-param>
    <param-name>db.password</param-name>
    <param-value>sa</param-value>
</context-param>
<context-param>
    <param-name>db.tcpServer</param-name>
    <param-value>-tcpAllowOthers</param-value>
</context-param>
```

当 web 应用停止时，数据库连接将被自动关闭，如果通过 DbStarter 还启动了 TCP 服务，TCP 服务也将被自动关闭。

使用 **H2** 控制台 **Servlet**

H2 控制台是一个包含在 web 服务中的独立的应用，但是它也能作为一个servlet使用。为了做到这点，你需要将`h2*.jar`文件添加到你的应用中，在你的 web.xml 文件中增加下面的配置：

```
<servlet>
    <servlet-name>H2Console</servlet-name>
    <servlet-class>org.h2.server.web.WebServlet</servlet-class>
    <!--
    <init-param>
        <param-name>webAllowOthers</param-name>
        <param-value></param-value>
    </init-param>
    <init-param>
        <param-name>trace</param-name>
        <param-value></param-value>
    </init-param>
    -->
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>H2Console</servlet-name>
    <url-pattern>/console/*</url-pattern>
</servlet-mapping>
```

关于更多的细节，请参考`src/tools/WEB-INF/web.xml`

要创建一个合适的 H2 控制台的 web 应用，运行下面的命令：

```
build warConsole
```

Android

Android 设备(使用Dalvik VM)上可以使用这个数据库代替 SQLite。到目前为止,只有很少的测试和基准测试运行,但似乎性能类似于 SQLite,除了打开和关闭数据库没有优化 (H2大约需要0.2秒,和SQLite约0.02秒)。读操作似乎比 SQLite 快,而写操作慢点。到目前为止,只有很少的测试运行,一切似乎正常工作。全文搜索还没有测试,然而本地的全文搜索应该能工作。

使用 H 2而不是 SQLite 原因是:

- 全 Unicode 支持包括 UPPER() 和 LOWER()
- 流 API 用于 BLOB 和 CLOB 数据
- 全文搜索
- 多连接
- 用于定义方法和触发器
- 数据库文件加密
- 读写 CSV 文件 (该功能也可以在数据库外部使用)
- 引用完整性和检查约束
- 更好的数据类型和 SQL 支持
- 内存数据库、只读数据库、表关联
- 与其他数据库更好的兼容,简化应用程序移植
- 可能更好的性能（对读操作）
- 服务器模式(在不同的机器上通过 TCP/IP 访问同一个数据库)。

目前只支持 JDBC API (计划在将来的版本中支持 Android 数据库API)。常规的 H2 jar文件和小 `h2small-*.jar` 可以使用。为了创建更小的 jar 文件,运行命令 `./build.sh jarSmall` (Linux / Mac OS) 或 `build.bat jarSmall` (Windows)

数据库文件需要存储在一个可以被一个应用程序访问的地方。例子:

```
String url = "jdbc:h2:/data/data/" +
    "com.example.hello" +
    "/data/hello" +
    ";FILE_LOCK=FS" +
    ";PAGE_SIZE=1024" +
    ";CACHE_SIZE=8192";
Class.forName("org.h2.Driver");
conn = DriverManager.getConnection(url);
...
```

限制:使用连接池目前不支持,因为所需的javax.sql 类是不在 Android 上的。

CSV (Comma Separated Values) 支持

CSV（逗号分隔文件）文件在数据库系统中支持 `CSVREAD` 和 `CSVWRITE` 方法，也可以把它作为数据库之外的一个工具来使用。将数据库查询结果写成CSV文件

通过数据库读取 CSV 文件

使用 `CSVREAD`，例子：

```
SELECT * FROM CSVREAD('test.csv');
```

请注意由于性能原因, `CSVREAD` 不应使用在 jion 内部。相反,先导入数据(可能到一个临时表),如果有必要创建所需的索引,然后查询这个表。

通过 CSV 文件导入数据

从CSV文件快速加载或导入数据(有时称为“批量加载”)的方法是结合表创建和导入。可选地,该列名称和数据类型可以在创建表时设置。另一个选项 `INSERT INTO ... SELECT`。

```
CREATE TABLE TEST AS SELECT * FROM CSVREAD('test.csv');
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255))
    AS SELECT * FROM CSVREAD('test.csv');
```

通过数据库写 CSV 文件

使用 `CSVWRITE`，

```
CREATE TABLE TEST(ID INT, NAME VARCHAR);
INSERT INTO TEST VALUES(1, 'Hello'), (2, 'World');
CALL CSVWRITE('test.csv', 'SELECT * FROM TEST');
```

通过 Java 应用写 CSV 文件

使用 Csv 工具而无需数据库，

```
import java.sql.*;
import org.h2.tools.Csv;
import org.h2.tools.SimpleResultSet;
public class TestCsv {
    public static void main(String[] args) throws Exception {
        SimpleResultSet rs = new SimpleResultSet();
        rs.addColumn("NAME", Types.VARCHAR, 255, 0);
        rs.addColumn("EMAIL", Types.VARCHAR, 255, 0);
        rs.addRow("Bob Meier", "bob.meier@abcde.abc");
        rs.addRow("John Jones", "john.jones@abcde.abc");
        new Csv().write("data/test.csv", rs, null);
    }
}
```

通过 Java 应用读 CSV 文件

读 CSV 文件可以无需打开数据库，

```
import java.sql.*;
import org.h2.tools.Csv;
public class TestCsv {
    public static void main(String[] args) throws Exception {
        ResultSet rs = new Csv().read("data/test.csv", null, null);
        ResultSetMetaData meta = rs.getMetaData();
        while (rs.next()) {
            for (int i = 0; i < meta.getColumnCount(); i++) {
                System.out.println(
                    meta.getColumnLabel(i + 1) + ": " +
                    rs.getString(i + 1));
            }
            System.out.println();
        }
        rs.close();
    }
}
```

升级、备份和恢复

数据库升级

数据库升级的推荐方案是，老版本的数据库的数据备份成 SQL 脚本的方式，在新版本的数据库上执行这些 SQL 来恢复数据。

使用 **Script** 工具备份数据

备份数据库有多种方式。如可以直接拷贝数据库文件，但是不建议在数据库在使用的时候去拷贝文件，另外数据库文件是二进制的，不能直接读懂，并且数据库文件可能会比较大，推荐的备份方式是创建压缩的 SQL 脚本文件，并且 H2 提供了数据导出的 `Script` 工具：

```
java org.h2.tools.Script -url jdbc:h2:~/test -user sa -script test.zip -options compression zip
```

也可能通过 SQL 命令 `SCRIPT` 去备份数据库，关于更多的命令选项，请查看 SQL 命令 `SCRIPT`。备份也能通过远程来做，但是文件被创建在服务器上，你可以通过 FTP 服务获取备份的脚本文件。

用脚本恢复数据

从一个SQL脚本文件恢复数据库，你可以使用 `RunScript` 工具：

```
java org.h2.tools.RunScript -url jdbc:h2:~/test -user sa -script test.zip -options compression zip
```

关于更多的命令的选项，请参考 SQL 命令 `RUNSCRIPT`。恢复也能通过远程来实现，但是恢复的文件需要在服务器上。可以通过FTP服务器上传恢复需要的脚本文件。也可以通过 `RUNSCRIPT` 执行 SQL 脚本，SQL 脚本文件内也可以引用另外的 SQL 脚本文件，在服务器模式下，也可以远程执行 SQL 脚本，但是要求脚本文件和被引用的脚本未见都在服务器上。

在线备份

`BACKUP` SQL 语句和 `Backup` 工具都能创建全库的备份文件的压缩 zip 包。但是，这个文件的内容并不可读

相比脚本方式，BACKUP命令并不锁定数据库对象，也不阻塞用户，但是 BACKUP 命令备份结果是事务一致的。：

```
BACKUP TO 'backup.zip'
```

`Backup` 工具 (org.h2.tools.Backup)不能创建在线备份；程序在执行的时候，数据库不能使用。

并不支持数据库运行的同时创建数据库备份，除非是文件系统支持创建快照，但是快照不能保证数据拷贝顺序的正确性。

命令行工具

H2数据库提供了一组命令行工具，如果你需要了解这些工具，使用参数 `-?`，如：

```
java -cp h2*.jar org.h2.tools.Backup -?
```

命令行工具有：

- Backup 创建数据库备份
- ChangeFileEncryption 允许改变文件加密密码和数据库的加密算法
- Console 启动基于浏览器的 H2 控制台
- ConvertTraceFile 转换 `.trace.db` 文件到 Java 应用和 SQL 脚本
- CreateCluster 从一个独立的数据库服务创建集群
- DeleteDbFiles 删除所有的数据库文件
- Recover 恢复损坏的数据库
- Restore 从数据库备份中恢复数据库
- RunScript 运行数据库 SQL 脚本
- Script 为数据库备份或迁移导出 SQL 脚本
- Server 启动 H2 服务模式
- Shell 命令行工具

这些工具也能在程序中通过调用相应的方法来使用，相关详细的调用说明，请参考 JavaDoc 文档。

Shell 工具

Shell 工具是最简单的命令行工具，开始时，输入:

```
java -cp h2*.jar org.h2.tools.Shell
```

你会要求输入数据库URL、JDBC 驱动、用户名和密码。连接设置也可以作为命令行参数设置。连接后,你会得到的列表选项。内置命令不需要以分号结束,但只是执行 SQL 语句,需要以分号 `;` 作为结束。这允许输入多行语句:

```
sql> select * from test
...> where id = 0;
```

默认情况下,结果是打印成表。结果有许多列,考虑使用模式列表:

```
sql> list
Result list mode is now on
sql> select * from test;
ID  : 1
NAME: Hello

ID  : 2
NAME: World
(2 rows, 0 ms)
```

使用 OpenOffice 基础框架

OpenOffice.org 基础框架支持通过 JDBC 连接数据库。你也可以通过 OpenOffice 框架连接到 H2 数据库。首先将 JDBC 驱动增加到 OpenOffice 中，下面的步骤可以连接到H2数据库：

- 启动 OpenOffice Writer，进入[Tools], [Options]
- 确认你在 OpenOffice.org/Java 中选择了 JAVA 运行环境
- 单击 [Class Path...], [Add Archive...]
- 选择你的 h2 的 jar 文件（本机上的路径可以由你选择）
- 单击 [OK] (按要求的点击), 停止OpenOffice (包括 Quickstarter)
- 启动 OpenOffice 框架
- 连接到一个存在的数据库；选择 [JDBC]; [Next]
- 输入数据库URL，如： `jdbc:h2:~/test`
- 输入 JDBC 驱动类： `org.h2.Driver`

你可以访问存于用户当前目录的数据库。

使用 H2 数据通过 NeoOffice (去掉 X11 的 OpenOffice)：

- 在 NeoOffice，到[NeoOffice]， [Preferences]
- 在[NeoOffice]页下找到[Java]
- 单击[Class Path], [Add Archive...]
- 选择 h2 的 jar 文件（本地目录，任你选择合适的目录）
- 单击[OK]（根据需要），重启 NeoOffice

现在，你可以通过"Database Wizard"创建一新的数据库：

- 单击[File], [New], [Database]
- 选择[Connect to existing database]，并且选择[JDBC]，单击next
- 输入数据源的URL，例如：`jdbc:h2:~/test`
- JDBC驱动类： org.h2.Driver

其他的在 NeoOffice 中使用 H2 的方法：

- 将 H2 的 jar 包打包到一个扩展包中
- 在 NeoOffice 中作为扩展 Java 包进行安装

这个能通过使用 NetBeans OpenOffice 插件来创建。详细看Extensions Development

Java Web Start / JNLP

当使用 Java Web Start / JNLP（JAVA网络加载协议），允许访问标签必须被设置在 .jnlp 文件，并且 .jar 的应用文件必须被签名，否则，当你试着去写文件系统，下面的异常将会被抛出：`java.security.AccessControlException`：拒绝访问 (`java.io.FilePermission ... read`). 如访问标签：

```
<security>
    <all-permissions/>
</security>
```

使用连接池

如果 H2 的数据库已经打开，打开一个连接很快。如果要打开和关闭许多连接，使用连接池，可以提升性能。H2 包含了一个简单的连接池，它是基于Christian d'Heureuse 的 Mini Connection Pool Manager 。还有其他更复杂的开源连接池可以使用，如Apache Commons DBCP。H2 从内置连接池获取连接比使用 `DriverManager.getConnection()` 快两倍左右。内置的连接

池使用方法如下：

```
import java.sql.*;
import org.h2.jdbcx.JdbcConnectionPool;
public class Test {
    public static void main(String[] args) throws Exception {
        JdbcConnectionPool cp = JdbcConnectionPool.create(
            "jdbc:h2:~/test", "sa", "sa");
        for (int i = 0; i < args.length; i++) {
            Connection conn = cp.getConnection();
            conn.createStatement().execute(args[i]);
            conn.close();
        }
        cp.dispose();
    }
}
```

全文检索

H2 包含了两种全文检索的实现。一种通过使用 Apache Lucene 来实现，另一种是通过存储索引文件到数据库里的一张特殊表来实现的（私有实现）。

使用私有全文检索

通过下面的调用来实现初始化：

```
CREATE ALIAS IF NOT EXISTS FT_INIT FOR "org.h2.fulltext.FullText.init";
CALL FT_INIT();
```

如果你要用到私有全文检索，你需要在每个数据库里都初始化它。然后你能创建一张表用于全文检索的索引，如：

```
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR);
INSERT INTO TEST VALUES(1, 'Hello World');
CALL FT_CREATE_INDEX('PUBLIC', 'TEST', NULL);
```

PUBLIC 是 schema 的名字， TEST 是索引表名。字段名列表是可选的，在上面的例子中，所有的字段都被索引，索引的更新是实时的，使用下面的查询语句可以进行搜索：

```
SELECT * FROM FT_SEARCH('Hello', 0, 0);
```

下面的语句将得到一个指定内容的搜索结果集。

```
QUERY: "PUBLIC"."TEST" WHERE "ID"=1
```

删除表上的索引

```
CALL FT_DROP_INDEX('PUBLIC', 'TEST');
```

要得到原始的数据，需要使用 `FT_SEARCH_DATA('Hello', 0, 0);` 。结果包含字段 SCHEMA (schema名), TABLE (表名), COLUMNS (字段名数组), 和 KEYS (对象数组)。可以和表做连接，使用连接如下：`SELECT T.* FROM FT_SEARCH_DATA('Hello', 0, 0) FT, TEST T WHERE FT.TABLE='TEST' AND T.ID=FT.KEYS[0];`

你也能在 Java 应用中使用索引：

```
org.h2.fulltext.FullText.search(conn, text, limit, offset);
org.h2.fulltext.FullText.searchData(conn, text, limit, offset);
```

用 Lucene 实现的全文检索

使用 Lucene 实现的全文检索，你需要将 Lucene 加入到 classpath 中，并且处理相关的依赖，如果你使用 H2 控制台，你能增加 Lucene 的 jar 文件到环境变量 `H2DRIVERS` 或 `CLASSPATH` 中。通过下面的语句初始化 Lucene 全文检索：

```
CREATE ALIAS IF NOT EXISTS FTL_INIT FOR "org.h2.fulltext.FullTextLucene.init";
CALL FTL_INIT();
```

如果你需要是使用 Lucene 全文检索，你每个数据库都需要初始化。你也能通过下面的语句创建全文索引表：

```
CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR);
INSERT INTO TEST VALUES(1, 'Hello World');
CALL FTL_CREATE_INDEX('PUBLIC', 'TEST', NULL);
```

PUBLIC 是 schema 名, TEST 是表名. 字段名列表是可选的， 在上面的例子中，所有的字段都被索引，索引的更新是实时的，使用下面的查询语句可以进行搜索：

```
SELECT * FROM FTL_SEARCH('Hello', 0, 0);
```

下面的语句将得到一个指定内容的搜索结果集：

```
QUERY: "PUBLIC"."TEST" WHERE "ID"=1
```

表上的索引(警告称,这将对所有的整个数据库的全文索引进行检索):

```
CALL FTL_DROP_INDEX('PUBLIC', 'TEST');
```

要得到原始的数据，需要使用 `FTL_SEARCH_DATA('Hello', 0, 0);`。结果包含字段 SCHEMA (schema名), TABLE (表名), COLUMNS (字段名数组), 和KEYS (对象数组)。可以和表做连接，使用连接如下 `SELECT T.* FROM FTL_SEARCH_DATA('Hello', 0, 0) FT, TEST T WHERE FT.TABLE='TEST' AND T.ID=FT.KEYS[0]`；

你也能在JAVA应用中使用索引：

```
org.h2.fulltext.FullTextLucene.search(conn, text, limit, offset);
org.h2.fulltext.FullTextLucene.searchData(conn, text, limit, offset);
```

Lucene 搜索支持全文搜索仅在特定的列。列名必须大写(如果原始列双援引除外)。为列名下划线(_),另一个强调需要添加。例子:

```
CREATE ALIAS IF NOT EXISTS FTL_INIT FOR "org.h2.fulltext.FullTextLucene.init";
CALL FTL_INIT();
DROP TABLE IF EXISTS TEST;
CREATE TABLE TEST(ID INT PRIMARY KEY, FIRST_NAME VARCHAR, LAST_NAME VARCHAR);
```

```
CALL FTL_CREATE_INDEX('PUBLIC', 'TEST', NULL);
INSERT INTO TEST VALUES(1, 'John', 'Wayne');
INSERT INTO TEST VALUES(2, 'Elton', 'John');
SELECT * FROM FTL_SEARCH_DATA('John', 0, 0);
SELECT * FROM FTL_SEARCH_DATA('LAST_NAME:John', 0, 0);
CALL FTL_DROP_ALL();
```

Lucene 全文搜索实现内部并不是同步的。如果你同时更新数据库和查询全文搜索(直接使用 Java API 的 H2 或 Lucene 本身),您需要确保操作正确同步。如果不是这样的话,你可能会有异常 `org.apache.lucene.store.AlreadyClosedException: this IndexReader is closed`。

用户定义变量

数据库支持用户自定义变量，自定义变量使用 `@` 开头，能够被用于任何表达式和参数中。变量是不能持久的，作为范围为 session，这就意味着变量只在定义它的那个 session 里是有效的。一个变量通常使用 SET命 令来声明：

```
SET @USER = 'Joe';
```

变量也可以通过使用 SET() 方法来改变值。在查询中可以直接使用：

```
SET @TOTAL = NULL;
SELECT X, SET(@TOTAL, IFNULL(@TOTAL, 1.) * X) F FROM SYSTEM_RANGE(1, 50);
```

变量不能被设置为 NULL 值，变量的类型是变量自动分配的，也就是说，在变量使用前变量的定义并不是必须（也不是必需）的，在声明变量时没有限制，大对象（LOB）也被支持。

日期和时间

日期、时间、时间戳的值支持ISO 8601格式，格式还包含了时区：

```
CALL TIMESTAMP '2008-01-01 12:00:00+01:00';
```

如果未设置时区，将使用系统中当前的时区。日期和时间信息存储在 H2 数据库文件,不包含时区信息。如果数据库打开使用另一个系统时区,日期和时间将是相同的。这意味着如果你存储的值“2000-01-01 12:00:00”在一个时区,然后再关闭数据库并打开数据库在不同的时区,你还将得到“2000-01-01 12:00:00”。请注意,更改时区 H2 加载驱动程序不支持。

使用 Spring

使用 TCP 服务器

在 Spring 配置如下，来启动和关闭 H2 TCP 服务器

```
<bean id = "org.h2.tools.Server"
            class="org.h2.tools.Server"
            factory-method="createTcpServer"
            init-method="start"
            destroy-method="stop">
    <constructor-arg value="-tcp,-tcpAllowOthers,-tcpPort,8043" />
</bean>
```

destroy-method 方法能防止热重部署和重启异常。

错误代码不兼容

有一个不相容，就是 Spring JdbcTemplate 和 H2 version 1.3.154版本及更新的版本,因为错误代码有变化。这将导致 JdbcTemplate 不是检测重复的关键条件,所以一个 DataIntegrityViolationException 代替 DuplicateKeyException 抛出。参见 SPR-8235 的问题。解决方法是添加以下 XML 文件的根路径:

```
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
        "http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd"
    >
    <import resource="classpath:org/springframework/jdbc/support/sql-error-codes.xml"/>
    <bean id = "H2" class="org.springframework.jdbc.support.SQLErrorCodes">
        <property name="badSqlGrammarCodes">
            <value>
                42000,42001,42101,42102,42111,42112,42121,42122,42132
            </value>
        </property>
        <property name="duplicateKeyCodes">
            <value>23001,23505</value>
        </property>
        <property name="dataIntegrityViolationCodes">
            <value>22003,22012,22025,23000</value>
        </property>
        <property name="dataAccessResourceFailureCodes">
            <value>90046,90100,90117,90121,90126</value>
        </property>
        <property name="cannotAcquireLockCodes">
            <value>50200</value>
        </property>
    </bean>
</beans>
```

OSGi

标准的 H2 jar 可以下降为一捆在一个 OSGi 容器中。H2 实现了 OSGi Service Platform Release 4 Version 4.2 Enterprise Specification 中定义 JDBC 服务 。H2 数据源工厂服务注册以下属性: `OSGI_JDBC_DRIVER_CLASS=org.h2.Driver` 和 `OSGI_JDBC_DRIVER_NAME=H2` 。 `OSGI_JDBC_DRIVER_VERSION` 属性反映了版本的驱动程序。

支持以下标准配置属性: `JDBC_USER, JDBC_PASSWORD, JDBC_DESCRIPTION, JDBC_DATASOURCE_NAME, JDBC_NETWORK_PROTOCOL, JDBC_URL, JDBC_SERVER_NAME, JDBC_PORT_NUMBER` 。任何其他标准属性将被拒绝。非标准的属性将被传递给 H2 连接的 URL

Java Management Extension (JMX)

支持 JMX 管理, 但不是默认启用。为了启用 JMX,追加 `;JMX=TRUE` 到数据库 URL， 当数据库打开时。各种工具支持 JMX,其中一个是 `jconsole` 工具。当打开 `jconsole` ,可以连接到打开的数据库的进程(使用服务器模式时,您需要连接到服务器进程)。然后去MBeans 部分。 在 `org.h2` 下每个数据库都会发现一个实体。这个实体对象名称的是数据库短名称,加上路径(每个冒号被替换为下划线字符)。

以下属性和操作的支持:

- CacheSize:当前使用的缓存大小， 单位KB
- CacheSizeMax(读/写):最大缓存大小， 单位KB
- Exclusive:这个数据库是否在独占模式
- FileReadCount:数据库打开后，文件读取操作的数量
- FileSize:文件大小， 单位KB
- FileWriteCount:数据库打开后，文件写操作的数量
- FileWriteCountTotal:数据库创建后，文件写操作的数量
- LogMode(读/写):当前事务日志模式。有关详细信息,请参阅 `SET LOG`
- Mode:兼容性模式(`REGULAR` 如果没有使用兼容模式)
- MultiThreaded:如果启用了多线程就是 true
- Mvcc:如果启用了Mvcc 就是 true
- ReadOnly:如果数据库是只读的就是 true
- TraceLevel (读/写):文件跟踪级别
- Version:使用的数据库版本
- listSettings:列出数据库设置
- listSessions:开放会话列表,包括当前执行语句(如果有的话)和锁定的表(如果有的话)

要启用 JMX,可能需要设置 JVM 系统属性 `com.sun.management.jmxremote` 和 `com.sun.management.jmxremote.port`

特性

特性列表

主要特性

- 非常快的数据库引擎
- 开源
- Java 编写
- 支持标准 SQL, JDBC API
- 内嵌和服务器模式，支持集群
- 强大的安全特性
- 可使用 PostgreSQL ODBC
- 多版本并发

其他特性

- 基于磁盘或者内存数据库和表，支持只读数据库，临时表
- 支持事务（read committed）,两阶段提交
- 多连接，表级锁
-

Comparison to Other Database Engines

H2 in Use

Connection Modes

Database URL Overview

Connecting to an Embedded (Local) Database

In-Memory Databases

Database Files Encryption

Database File Locking

Opening a Database Only if it Already Exists

Closing a Database

Ignore Unknown Settings

Changing Other Settings when Opening a Connection

Custom File Access Mode

Multiple Connections

Database File Layout

Logging and Recovery

Compatibility

Auto-Reconnect

Automatic Mixed Mode

Page Size

Using the Trace Options

Using Other Logging APIs

Read Only Databases

Read Only Databases in Zip or Jar File

Computed Columns / Function Based Index

Multi-Dimensional Indexes

User-Defined Functions and Stored Procedures

Pluggable or User-Defined Tables

Triggers

Compacting a Database

Cache Settings

入门

欢迎使用 H2。 H2 是一个 Java SQL 数据库。H2 的主要特点是:

- 运行很快,开源,支持 JDBC API
- 支持嵌入模式和服务器模式;是一个内存数据库
- 基于浏览器控制台应用程序
- 文件很小大，jar文件约 1.5 MB

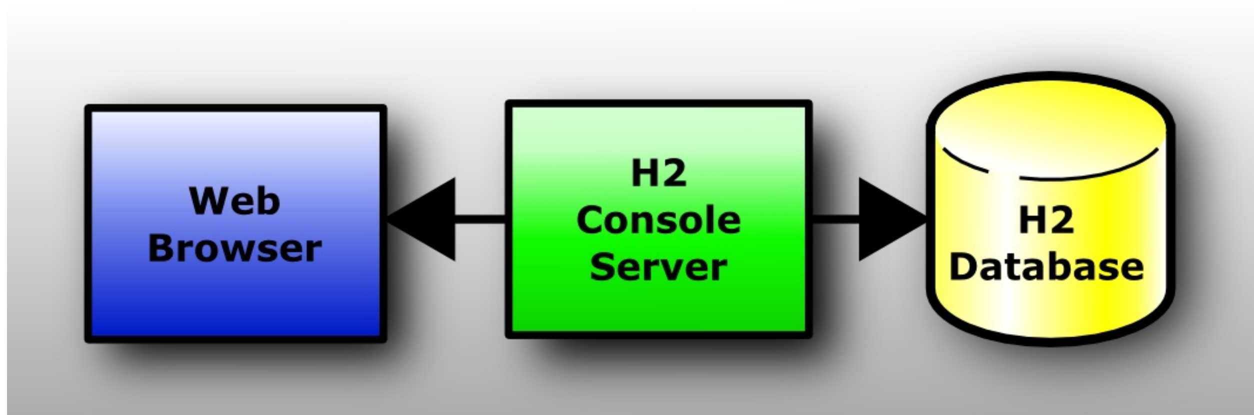
嵌入 H2 到 应用中

支持嵌入模式和服务器模式。若使用嵌入模式，需做如下步骤：

- 添加 `h2*.jar` 到 classpath (H2 没有任何依赖)
- 使用 JDBC 驱动类 : org.h2.Driver
- 数据库的 URL是 `jdbc:h2:~/test` ,在你的用户目录打开数据库 `test`
- 这样新的数据库就自定创建了

H2 控制台程序

控制台允许你通过浏览器访问 SQL 数据库



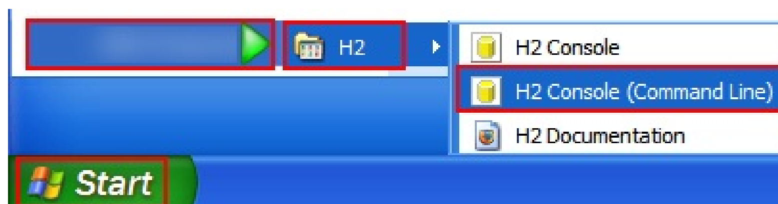
下面演示步骤是 Windows 平台。

安装

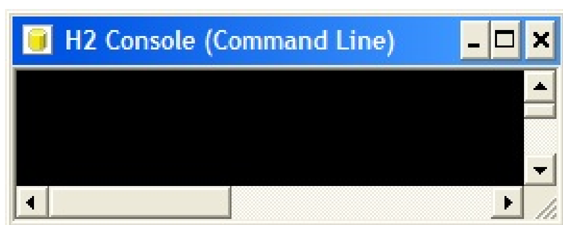
使用 Windows Installer（可以在 http://www.h2database.com/html/download.html 下载）

启动控制台

点击 [Start], [All Programs], [H2], 和 [H2 Console (Command Line)]:



此时就能看到控制台界面



同时，一个浏览器窗口将被打开，指向的URL是http://localhost:8082，是 H2 管理系统的登录页面。有可能会有一个防火墙的提示信息，如果不设置防火墙，将只能在本地访问。

登陆

选择 [Generic H2] 并点击 [Connect]:

English Preferences Help

Login

Saved Settings: Generic H2

Setting Name: Generic H2 Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:test

User Name: sa

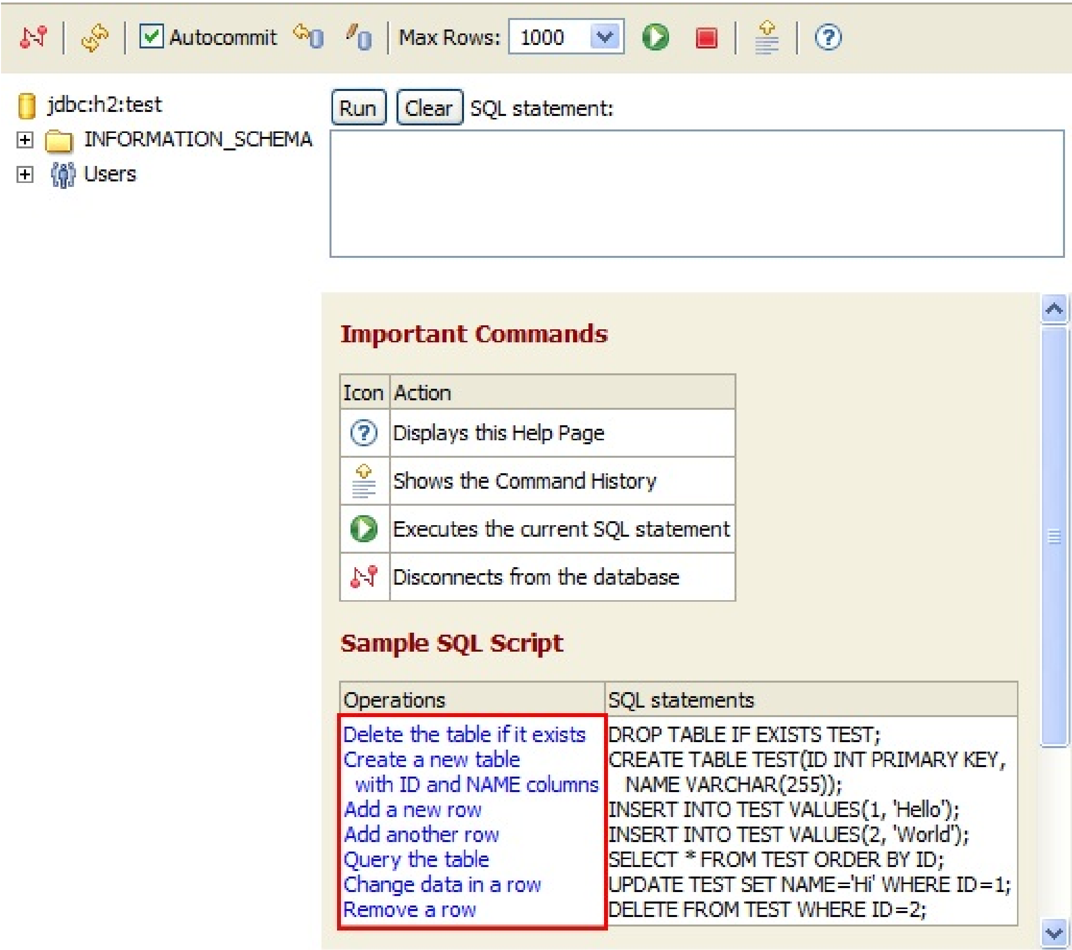
Password:

Connect Test Connection

这样你就登陆了

示例

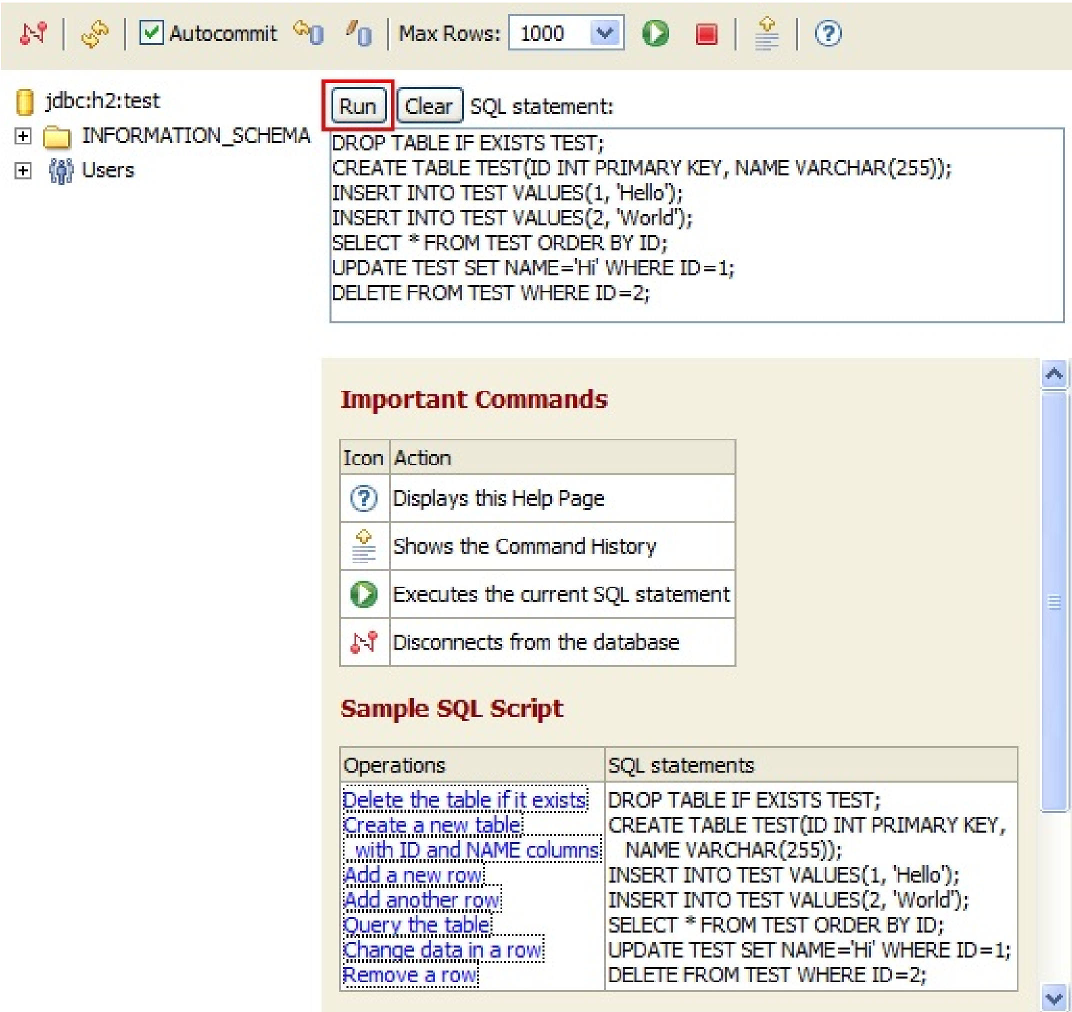
点击 [Sample SQL Script]:



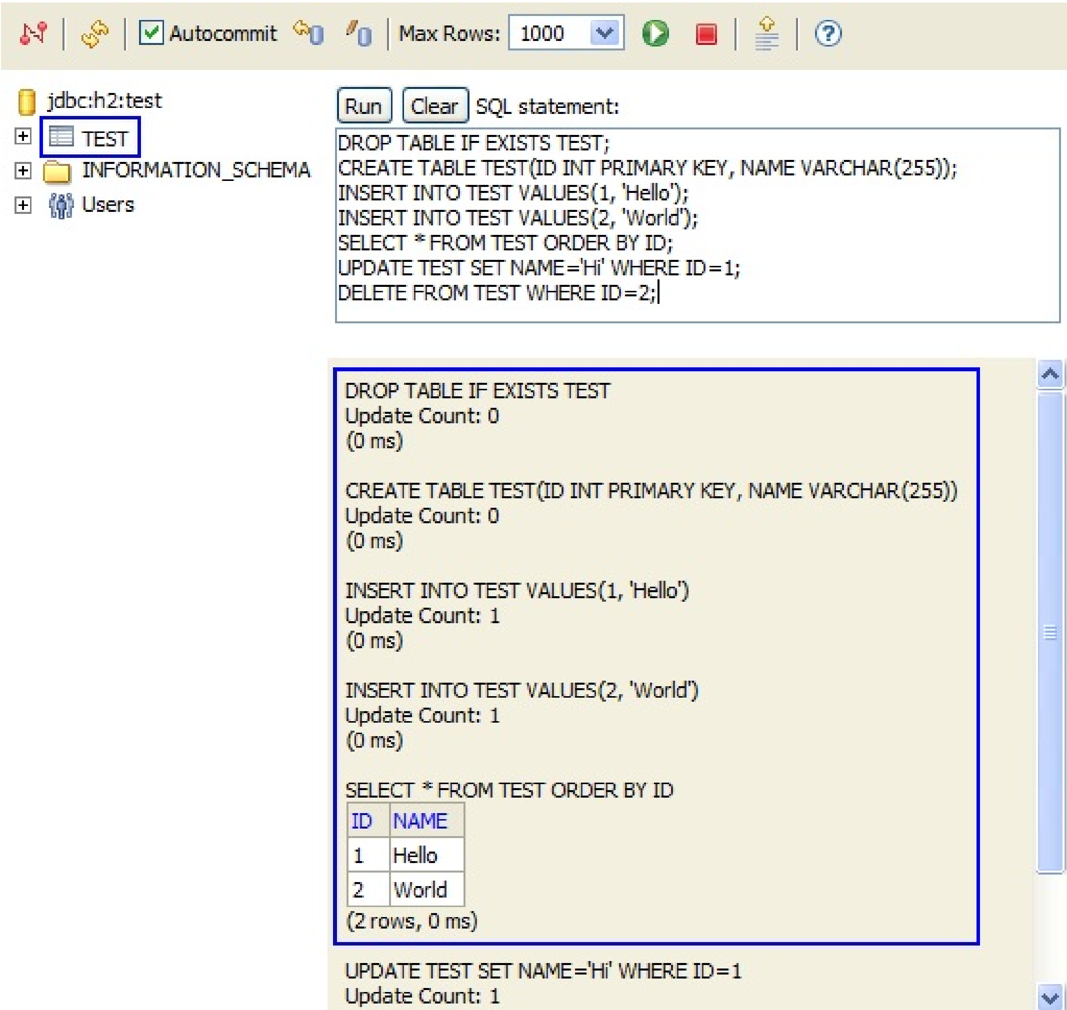
SQL 命令出现在命令行区域

执行

点击 [Run]



在左侧，可以看到一个新的 TEST 被添加。操作的结果显示如下脚本：



断开

点击 [Disconnect]



来断开连接

关闭

关闭控制台窗口即可。

更详细的步骤说明，可以参阅教程一章